НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

«На правах рукопису»
УДК 004.043

«До захисту допущено»
Завідувач кафедри
_____ І.Р. Пархомей
(підпис)
“___”_____2019 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: ____Інтелектуальна система розпізнавання видів грибів

Виконав: студент другого курсу, групи ІТ-84мп
(шифр групи)

Голіков Назар Андрійович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент, к.т.н., доцент Корнага Я.І. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____НК_____ ____к.т.н., доцент, Пасько В.П. _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент __д.т.н., проф. , професор каф. ММСА, Мухін В.Є._____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

Рівень вищої освіти – другий (магістерський)

Спеціальність 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____І.Р. Пархомей
(підпис)
«___»_____2019 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Голікову Назару Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації «Інтелектуальна система розпізнавання видів грибів»,
_____
науковий керівник дисертації__доцент, к.т.н., доцент Корнага Я. І.,_____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «___»_____ 2019 р. №___

2. Термін подання студентом дисертації _____

3. Об'єкт дослідження – комп'ютерне бачення для розпізнавання грибів за даними зображення.

4. Предмет дослідження – розпізнавання видів грибів.

5. Перелік завдань, які потрібно розробити – аналіз проблеми та існуючих рішень; аналіз і реалізація алгоритму; розробка програмного забезпечення; дослідження ефективності розробленого моделі програмного забезпечення.

6. Орієнтовний перелік ілюстративного матеріалу – три плакати

7. Орієнтовний перелік публікацій – одна публікація

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз предметної області	20.09.2019 р.	
2	Постановка задачі	30.09.2019 р.	
3	Аналіз інформаційного забезпечення	15.10.2019 р.	
5	Аналіз алгоритмічного забезпечення	28.10.2019 р.	
6	Розробка алгоритмічного забезпечення	10.11.2019 р.	
7	Розробка програмного забезпечення	25.11.2019 р.	
8	Маркетинговий аналіз стартап-проекту	10.12.2019 р.	
9	Висновки	15.12.2019 р.	

Студент _____ Н. А. Голіков
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Я. І. Корнага
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

У роботі розглянуто проблему в області автоматизованого розпізнавання видів грибів на зображенні, показано основні особливості існуючих рішень та додатків, їх переваги та недоліки.

Для розпізнавання виду гриба за даними зображення розроблена нейронна мережа з розпізнавання. Для взаємодії користувача з системою розроблено програму бот месенджера Telegram. Дана система дозволяє зменшити імовірність помилкової ідентифікації грибів, а отже і кількість отруєнь грибами.

Визначено завдання для системи розпізнавання видів грибів на зображенні за допомогою нейронної мережі та відібрано нейронну мережу та спосіб навчання, які найбільш підходять для даної задачі. Описано структуру системи та проведено експерименти ефективності її роботи.

Ключові слова: нейронна мережа, машинне навчання, інтелектуальна система, морфологічні ознаки, бот.

Розмір пояснювальної записки – 80 аркушів, містить 23 ілюстрації, 27 таблиць, 5 додатків.

ABSTRACT

The paper deals with the problem in the field of automated recognition of mushroom species in the image, shows the main features of existing solutions and applications, their advantages and disadvantages.

To recognize the type of fungus according to the image developed neural network for recognition. A Telegram messenger bot program was developed for user interaction with the system. This system allows to reduce the likelihood of false identification of mushrooms, and therefore the number of mushroom poisoning.

The tasks for the system of recognition of mushroom species on the image using the neural network are determined and the neural network and the method of training that are most suitable for this task are selected. The structure of the system is described and experiments of its work efficiency are conducted.

Keywords: neural network, machine learning, intellectual system, morphological features, bot.

The size of the explanatory note - 80 sheets, contains 23 illustrations, 28 tables, 5 appendices.

Пояснювальна записка

до магістерської дисертації

на тему: *Інтелектуальна система розпізнавання видів грибів*

Київ – 2019 року

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

kNN – k nearest neighbors

HTTPS - HyperText Transfer Protocol Secure

API – Application Programming Interface

НМ – нейронна мережа;

СУБД – Система управління базами даних

HTML – HyperText Markup Language

БД – база даних

ПЗ – Програмне забезпечення

SQL – Structured Query Language

UML – Unified Modeling Language

ВСТУП

Інтелектуальні системи стають все більш популярними завдяки своїм величезним можливостям використання. Існує велика кількість завдань, в яких ефективніше використовувати саме інтелектуальні системи, оскільки вони можуть набагато швидше та якісніше виконувати спеціалізовані задачі. Основними плюсами існуючих в теперішній момент рішень є можливість автоматизації багатьох сфер діяльності при мінімізації участі при цьому людини та розширення сфер, де можна використовувати програмне забезпечення замість людської праці. Іншими словами інтелектуальна система – це штучний інтелект і наука про творчі системи та сама можливість створювати такі системи. Машинне навчання – це підрозділ штучного інтелекту, вивчаючий різні способи побудови алгоритмів, що навчаються. Під такими алгоритмами розуміються алгоритми, які змінюються (навчаються) якимось чином в залежності від вхідних даних. Машинне навчання – дуже велика область знань, однак, серед множини парадигм та підходів в машинному навчанні виділяється одна цікава область – штучні нейронні мережі. У даній статті буде розкрито роботу інтелектуальної системи розпізнавання видів грибів за допомогою нейронних мереж. Нейронні мережі черпають свою силу із розпаралелювання обробки інформації та із можливості самонавчатись, тобто створювати узагальнення. Під терміном узагальнення розуміється можливість отримувати аргументований результат на основі даних, які не зустрічались в процесі навчання. Ці властивості дозволяють нейронним мережам вирішувати масштабні задачі, які на сьогоднішній день є важковирішуваними. Окрім можливості вирішувати певний клас задач нейронні мережі мають ряд значних переваг. Всі плюси нейронних мереж являються наслідком плюсів біологічних нейронних мереж, так як сама модель обробки інформації практично не змінювалась. Загалом дисертація розкриває проблему отруєння грибами та один із способів її вирішення.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Постановка проблеми

Усім відомо, що в Україні існує проблема отруєння грибами. МОЗ України повідомляє, що останнім часом випадки отруєння грибами стали частішими, за статистикою в 2018 році отруїлось 174 людини, а також було зафіксовано 11 летальних випадків. Аналіз випадків отруєнь грибами свідчить, що більшість з них обумовлені вживанням пластинчастих отруйних грибів (у першу чергу блідої поганки), які помилково сприймаються за їстівні печериці та сироїжки. Нерідко отруюються грибами, які більшість досвідчених грибників спокійно кладуть до кошиків. Помилитися вони можуть, якщо збирають дуже молоді плодові тіла, коли ще не проявилися морфологічні ознаки. Наприклад, збирають сироїжки, а серед них може заховатися бліда поганка, яка ще не розкрилась і ззовні їх нагадує[1]. Тобто існує проблема, яка полягає в тому, що люди не завжди можуть правильно розізнати та класифікувати той чи інший вид грибів. Після вивчення проблематики отруєння грибами та аналізу можливих шляхів зменшення її рівня виникла ідея створення інтелектуальної системи, яка, використовуючи нейронну мережу, може класифікувати види грибів по фото для:

- своєчасного виявлення отруйних грибів;
- запобігання отруєння грибами;
- зменшення рівня летальних випадків;
- надання користувачам інформації щодо знайдених ними грибів.

1.2 Аналіз предметної області

У повсякденному житті ми називаємо грибами їх плодові тіла. У більшості їстівних грибів (за винятком трюфелів, сморжів та зморшків) плодове тіло утворено ніжкою і капелюшком, звідси й походить їх назва – шапинкові гриби.

Тіло шапинкових грибів – це міцелій, або грибниця, що утворений гіфами (рис. 1.1). Гіфи – це система дуже довгих та розгалужених мікроскопічних ниток, що утворюється в ґрунті.

Рисунок 1.1. Будова грибів

Ланцюжок видовжених безбарвних клітин утворюють гіфу, а гіфи – багатоклітинний міцелій. Якщо в тому місці, де знято гриб, злегка розрити ґрунт, можна виявити тонкі розгалужені білі нитки – грибницю. Грибниця – головна частина кожного гриба, вона у ґрунті з гіфами утворює величезну поверхню для вбирання необхідних речовин, а на поверхні ґрунту утворює плодові тіла. Капелюшок та ніжка складаються з щільно прилеглих один до одного ниток грибниці. У ніжці всі нитки однакові, а в капелюшку вони утворюють два шари – верхній, покритий шкіркою, пофарбованої різними пігментами, і нижній. У деяких грибів, таких як білий гриб, підберезник, нижній шар складається з численних трубочок, вони мають назву трубчасті, а в інших, наприклад, рижиків, нижній шар плодових тіл утворений численними пластинами, які розходяться від верхівки ніжки до краю шапинки, такі гриби називаються пластинчасті. Серед шапинкових грибів існують як їстівні так і отруйні гриби, розрізнити їх можна за морфологічними ознаками їх плодового тіла[2]. Наприклад, у найбільш відомих отруйних грибів є такі ознаки:

1. Бліда поганка:
   - тонка ніжка з «комірцем»;
   - бульбоподібне потовщення ніжки біля основи.
2. Несправжні опеньки:

- жовто-сіра в центрі іржавого кольору ніжка;
- без лусочок, жовтувато-зеленуваті пластини.

3. Жовчний гриб:
   - темний сітчастий малюнок на ніжці;
   - низ шапинки рожевий.
4. Мухомор:
   - червоний – має червону шапинку з білими плямами-лусочками, внизу ніжки потовщення, у верхній частині кільце;
   - пантерний – шапинка зеленувата або сіро-бура з маленькими білими лусочками, ніжка біла з кільцем у верхній частині та потовщенням внизу;
   - порфировий – схожий на печерицю, але ніжка тонка, лусочок на шапинці часто немає.

Надання біологічної класифікації гриба (тобто правильне визначення роду та виду грибів) вимагає звернути увагу на широкий спектр особливостей, багато з них помітні при звичайному огляді плодового тіла гриба, інші ж виявляються лише після мікроскопічного дослідження[3].

Машинне навчання – це наукове вивчення алгоритмів та статистичних моделей, які комп'ютерні системи використовують для виконання конкретного завдання без використання чітких інструкцій, покладаючись на закономірності та умовиводи. Машинне навчання розглядається як підмножина штучного інтелекту. Алгоритми машинного навчання будують математичну модель на основі вибіркових даних, відомих як "дані тренувань", щоб приймати прогнози чи рішення, не будучи явно запрограмованими для виконання завдання. Алгоритми машинного навчання використовуються в найрізноманітніших програмах, таких як фільтрування електронної пошти та комп’ютерний зір, де складно або нездійсненно розробити звичайний алгоритм для ефективного виконання завдання[4].

Машинне навчання тісно пов'язане з обчислювальною статистикою, яка зосереджена на прогнозуванні за допомогою комп'ютерів. Вивчення математичної оптимізації надає методи, теорії та області застосування в область

машинного навчання. Обмін даними – це сфера вивчення в рамках машинного навчання і фокусується на дослідницькому аналізі даних за допомогою непідконтрольного навчання.

Назву машинного навчання було придумано в 1959 році Артуром Самуелем. Том Мітчелл надав широко цитоване, більш формальне визначення алгоритмів, що вивчаються в галузі машинного навчання: "Кажуть, що комп'ютерна програма вивчає досвід E щодо деякого класу завдань T і міра ефективності P, якщо її виконання на завданнях в T, виміряне P, покращується з досвідом E. " Це визначення завдань, що стосуються машинного навчання, пропонує принципово оперативне визначення, а не визначення галузі в когнітивному розумінні[5].

Розрізняють два типи навчання:

- Навчання по прецедентах, або індуктивне навчання, засноване на виявленні емпіричних закономірностей в даних.
- Дедуктивне навчання передбачає формалізацію знань експертів і їх перенесення в комп'ютер у вигляді бази знань.

Дедуктивне навчання прийнято відносити до області експертних систем, тому терміни машинне навчання і навчання по прецедентах можна вважати синонімами.

Багато методів індуктивного навчання розроблялися як альтернатива класичним статистичним підходам. Багато методів тісно пов'язані зі вилученням інформації (information extraction), інтелектуальним аналізом даних (data mining)[6].

Загальна постановка задачі навчання по прецедентах.

Є безліч об'єктів (ситуацій) і безліч можливих відповідей (відгуків, реакцій). Існує деяка залежність між відповідями і об'єктами, але вона невідома. Відома тільки кінцева сукупність прецедентів – пар «об'єкт, відповідь», звана навчальної вибіркою. На основі цих даних потрібно відновити неявну залежність, тобто побудувати алгоритм, здатний для будь-якого можливого вхідного об'єкта видати досить точну класифікуючу відповідь. Ця залежність не обов'язково виражається аналітично, і тут нейромережі реалізують принцип емпірично формованого рішення. Важливою особливістю при цьому є здатність

навченою системою до узагальнення, тобто до адекватного відгуку на дані, що виходять за межі наявної навчальної вибірки. Для вимірювання точності відповідей вводиться оціночний функціонал якості.

Дана постановка є узагальненням класичних задач апроксимації функцій. У класичних задачах апроксимації об'єктами є дійсні числа або вектори. У реальних прикладних задачах вхідні дані про об'єкти можуть бути неповними, неточними, нечисловими, різнорідними. Ці особливості призводять до великої різноманітності методів машинного навчання[7].

Штучна нейронна мережа – математична модель, а також її програмна або апаратна реалізація, побудована за принципом організації та функціонування біологічних нейронних мереж – мереж нервових клітин живого організму. Це поняття виникло при вивченні процесів, що протікають в мозку, і при спробі змоделювати ці процеси. Першою такою спробою були нейронні мережі У. Маккалок і У. Піттса. Після розробки алгоритмів навчання одержувані моделі стали використовувати в практичних цілях: в задачах прогнозування, для розпізнавання образів, в задачах управління та ін[8].

НМ є системою з'єднаних і взаємодіючих між собою простих процесорів (штучних нейронів). Такі процесори зазвичай досить прості (особливо в порівнянні з процесорами, використовуваними в персональних комп'ютерах). Кожен процесор подібної мережі має справу тільки з сигналами, які він періодично отримує, і сигналами, які він періодично посилає іншим процесорам. І, тим не менше, будучи з'єднаними в досить велику мережу з керованою взаємодією, такі окремо прості процесори разом здатні виконувати досить складні завдання[9].

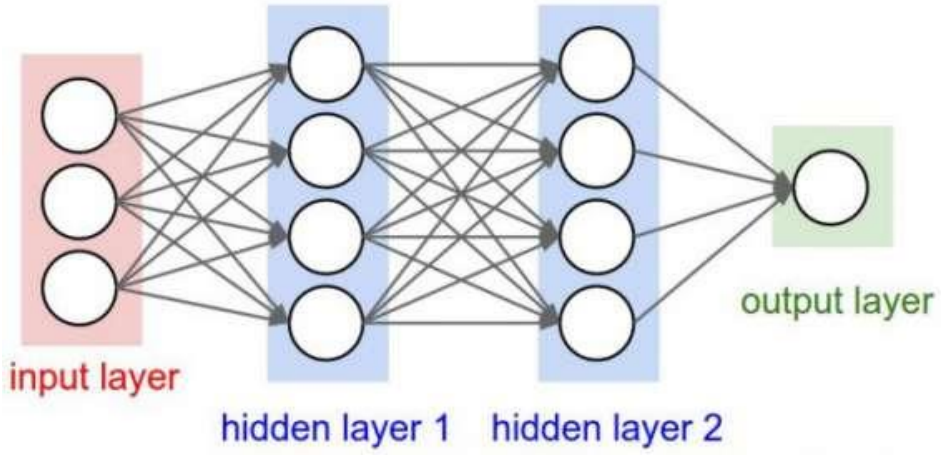


Рисунок 1.2. Схема простої нейронної мережі. Зеленим кольором позначені вхідні нейрони, блакитним – приховані нейрони, жовтим – вихідний нейрон

Способи машинного навчання

1. Навчання з учителем (supervised learning) – найбільш поширений випадок. Кожен прецедент являє собою пару «об'єкт, відповідь». Потрібно знайти функціональну залежність відповідей від описів об'єктів і побудувати алгоритм, який бере на вході опис об'єкта і видає на виході відповідь. Функціонал якості зазвичай визначається як середня помилка відповідей, виданих алгоритмом, по всіх об'єктах вибірки.

– Завдання класифікації (classification) відрізняється тим, що множина допустимих відповідей скінченна. Їх називають мітками класів (class label). Клас – це безліч всіх об'єктів з даними значенням мітки.

– Завдання регресії (regression) відрізняється тим, що допустимою відповіддю є дійсне число або числовий вектор.

– Завдання ранжирування (learning to rank) відрізняється тим, що відповіді треба отримати відразу на безлічі об'єктів, після чого впорядкувати їх за значеннями відповідей. Може зводитися до завдань класифікації або регресії. Часто застосовується в інформаційному пошуку і аналізі текстів.

– Завдання прогнозування (forecasting) відрізняється тим, що об'єктами є відрізки часових рядів, що обриваються в той момент, коли потрібно зробити прогноз на майбутнє. Для вирішення завдань прогнозування часто вдається пристосувати методи регресії або класифікації, причому в другому випадку мова йде скоріше про завдання прийняття рішень[10].

2. Навчання без вчителя (unsupervised learning). В цьому випадку відповіді не задаються, і потрібно шукати залежності між об'єктами.

– Завдання кластеризації (clustering) полягає в тому, щоб згрупувати об'єкти в кластери, використовуючи дані про попарну схожість об'єктів. Функціонали якості можуть визначатися по-різному, наприклад, як відношення середніх міжкластерних і внутрікластерних відстаней.

– Завдання пошуку асоціативних правил (association rules learning). Вихідні дані подаються у вигляді признакових описів. Потрібно знайти такі набори ознак, і такі значення цих ознак, які особливо часто зустрічаються в признакових описах об'єктів.

– Задача фільтрації викидів (outliers detection) – виявлення в навчальній вибірці невеликого числа нетипових об'єктів. У деяких додатках їх пошук є самоціллю (наприклад, виявлення шахрайства). В інших додатках ці об'єкти є наслідком помилок в даних або неточності моделі, тобто шумом, що заважає налаштовувати модель, і повинні бути видалені з вибірки.

– Завдання скорочення розмірності (dimensionality reduction) полягає в тому, щоб по вихідним ознаками за допомогою деяких функцій перетворення перейти до найменшого числа нових ознак, не втративши при цьому жодної суттєвої інформації про об'єкти вибірки. У класі лінійних перетворень найбільш відомим прикладом є метод головних компонент.

– Завдання заповнення пропущених значень (missing values) – заміна відсутніх значень в матриці об'єкти-ознаки їх прогнозними значеннями.

3. Часткове навчання (semi-supervised learning) займає проміжне положення між навчанням з учителем і без вчителя. Кожен прецедент являє собою пару «об'єкт, відповідь», але відповіді відомі тільки на частині прецедентів. Приклад прикладної задачі – автоматична рубрикація великої кількості текстів за умови, що деякі з них вже віднесені до якихось рубриках[11].

4. Трансдуктивне навчання (transductive learning). Дана кінцева навчальна вибірка прецедентів. Потрібно за цим приватним даними зробити передбачення относительно інших приватних даних – тестової вибірки. На відміну від стандартної постановки, тут не потрібно виявляти загальну закономірність, оскільки відомо, що нових тестових прецедентів не буде. З іншого боку, з'являється можливість поліпшити якість прогнозів за рахунок аналізу всієї тестової вибірки цілком, наприклад, шляхом її кластеризації. У багатьох додатках трансдуктівное навчання практично не відрізняється від часткового навчання.

5. Навчання з підкріпленням (reinforcement learning). Роль об'єктів грають пари «ситуація, прийняте рішення», відповідями є значення функціоналу якості, що характеризує правильність прийнятих рішень (реакцію середовища). Як і в задачах прогнозування, тут істотну роль грає фактор часу. Приклади прикладних задач: формування інвестиційних стратегій, автоматичне керування технологічними процесами, самонавчання роботів, і т.д.

6. Динамічне навчання (online learning) може бути як навчанням з учителем, так і без вчителя. Специфіка в тому, що прецеденти надходять потоком. Потрібно негайно приймати рішення по кожному прецеденту і одночасно доучувати модель залежності з урахуванням нових прецедентів. Як і в задачах прогнозування, тут істотну роль грає фактор часу.

7. Активне навчання (active learning) відрізняється тим, що учень має можливість самостійно призначати наступний прецедент, який стане відомий.

8. Многозадачне навчання (multi-task learning). Набір взаємопов'язаних або схожих завдань навчання вирішується одночасно, за допомогою різних алгоритмів навчання, що мають схожу внутрішньо уявлення. Інформація про подібність завдань між собою дозволяє більш ефективно вдосконалювати алгоритм навчання і підвищувати якість вирішення основного завдання.

1.3 Огляд існуючих аналогів

Проаналізувавши ринок систем розпізнавання видів грибів, було виявлено велику кількість компаній, що займаються розробкою та вдосконаленням таких систем, проте більшість з них мають велику кількість недоліків у роботі і не завжди надають корисну, а головне достовірну інформацію для користувачів.

1.3.1«Довідник грибника».

«Довідник грибника» (рис.1.3) – мобільний додаток для пристроїв, що працюють на операційній системі Android, створений компанією Haedcorp. Він містить інформацію про різні параметри гриба, місця його росту, період плодоношення, їстівність та інші корисні відомості. У додатку описано 314 видів грибів як їстівних, умовно-їстівних так і неїстівних, зібрано більше 1400 фотографій грибів з різних ракурсів, що дозволяє користувачу максимально точно ідентифікувати знайдений гриб.

Рисунок 1.3. Інтерфейс додатку «Довідник грибника»

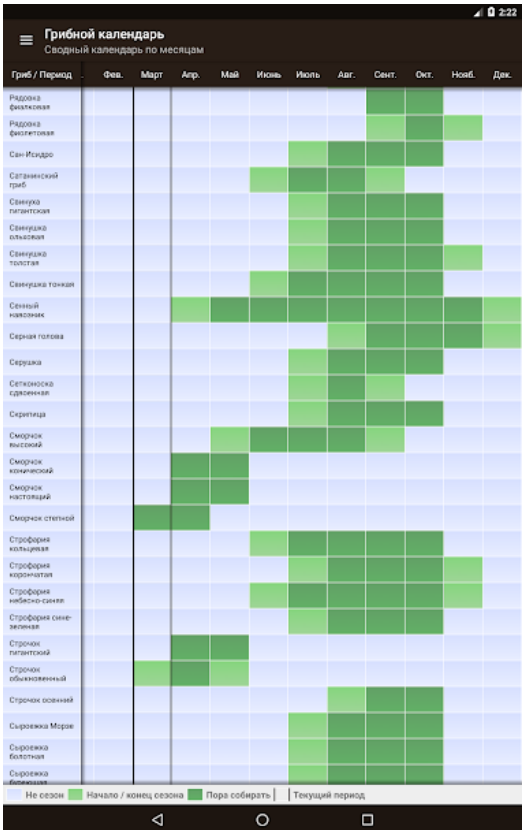


Рисунок 1.4. Грибний календар

Із плюсів можна виділити:

– можливість створення власного списку грибів, який можна швидко побачити в будь-який момент;

– розділ «Сезонність», що дозволяє дізнатись які саме гриби ростуть в даний момент, а також побачити зведений грибний календар по місяцях(рис.1.4);

– розділ «Статті про гриби» містить масу корисної інформації, яка допоможе початківцям грибникам зібратися в дорогу, а також не загубитися в лісі.

Основним недоліком даного додатку є те, що користувач, знайшовши гриб повинен сам знайти його серед великої кількості заявлених грибів, на такі пошуки може піти багато часу. Користувачами також зазначалося, що інформація наведена про гриби уже застаріла, фото не завжди якісні, а деякі види віднесені не до своєї категорії, тому користуватись таким додатком не завжди безпечно. Також можна виділити те, що додаток не кросплатформений.

1.3.2 «Розпізнавання грибів по фото»

«Розпізнавання грибів по фото»(рис.1.5) – мобільний додаток для пристроїв, що працюють на операційній системі Android, створений компанією Dominik Steinhauser, що дозволяє визначити назву та тип гриба, завантаживши його фото з різних ракурсів.

Серед плюсів виділяється:

– простота у використанні;

– швидкість розпізнавання;

– велика варіативність.

Рисунок 1.5. Інтерфейс додатку «Розпізнавання грибів по фото»

Найбільшим мінусом даного аналогу є дуже низька точність розпізнавання грибів, вони неправильно класифікуються, що може призвести до фатальних наслідків. Також цей додаток не є кросплатформеним.

1.3.3 «Гриби: велика енциклопедія»

«Гриби: велика енциклопедія»(рис.1.6) – мобільний додаток для пристроїв, що працюють на операційній системі ios, створений компанією AppGrade. Він містить детальний опис з зображеннями і фотографіями грибів, а також теоретичну частину і поради, які будуть корисні як грибникам-початківцям так і грибникам зі стажем. Додаток містить необхідні для роботи функції, має зручний та звичний інтерфейс в поєднані з унікальним дизайном:

– Не потребує підключення до інтернету;

– Режим перегляду списку «обкладинка альбому» або «сітка» спростить швидкий пошук гриба по його зовнішньому вигляді;

– Функція створення фотонотаток з збереженням координат дозволить розширити базу власними фотографіями грибів або створення власної карти грибних місць;

– Налаштування інтерфейсу з урахуванням індивідуальних потреб;

– «Вибране» і «історія» дозволяють вибрати найважливіше про гриби;

– Можливість відкрити опис гриба у «Вікіпедії» (якщо він там є) не покидаючи при цьому додаток.

Рисунок 1.5. Інтерфейс додатку «Гриби: велика енциклопедія»

Рисунок 1.6. Корисна інформація про гриби

Також серед плюсів виділяються наступні:

– Більше 180 видів грибів, поділених заради зручності навігації на категорії і характерні кольори;

– Більше 950 фотографій і ілюстрацій(рис.1.6);

– Підказки як не переплутати їстівний гриб з отруйним;

– «Розумний наскрізний пошук» по всій базі спільно з підсвічуванням знайдених слів, дозволяє оперативно знайти потрібний гриб по його опису;

– Ведеться історія записів.

У процесі експлуатації було виявлено ряд недоліків:

– Працює нестабільно, на одне й те ж саме фото видає різний опис;

– Неправильне розпізнавання грибів;

– Під час читання інформації про гриби, при перегортувані сторінок додаток «висне»;

– Не є кросплатформеним.

Висновки до розділу

В розділі проведено постановку проблеми із зазначенням її актуальності на даний час. Також здійснено огляд предметної області, огляд підходів до вирішення проблеми та аналіз сучасних аналогів з метою виявлення їх сильних та слабких сторін.

РОЗДІЛ 2 ПОРІВНЯННЯ СУЧАСНИХ ПІДХОДІВ ТА ІНСТРУМЕНТІВ

2.1 Огляд платформи Telegram Bots

Telegram – безкоштовний месенджер, який дозволяє надсилати повідомлення, файли, медіа, робити дзвінки. За допомогою спеціального API можна створювати програми названі «ботами». Введемо термін «бот» – бот це програма, яка є інтегрованою з месенджером «Telegram». Користувачі можуть взаємодіяти з нею шляхом посилання повідомлень, медіа файлів, локації, команд. У 2015 році було запущено цей сервіс і зараз можна знайти ботів на різну тематику, таку як: замовлення таксі відправивши свою локацію, боти мереж новин, боти інтернет магазинів, боти перекладачі і т.д. Вся взаємодія відбувається використовуючи HTTPS запити(рис. 2.1).

Взаємодія з ботом виконується наступним чином:

- Користувач відправляє боту повідомлення;
- Бот, використовуючи технологію Long Polling або Webhook, отримує; від сервера Telegram повідомлення користувача;
- Бот генерує відповідь на повідомлення;
- Бот відправляє серверу Telegram відповідь;
- Сервер Telegram відправляє користувачу повідомлення від бота.

Фактично сервер Telegram є посередником між користувачами та ботом. Telegram ідентифікує бота використовуючи унікальний токен, який необхідно отримати у бота, який керує ботами. Цей токен використовується у всіх запитах, що бот надсилає серверу.

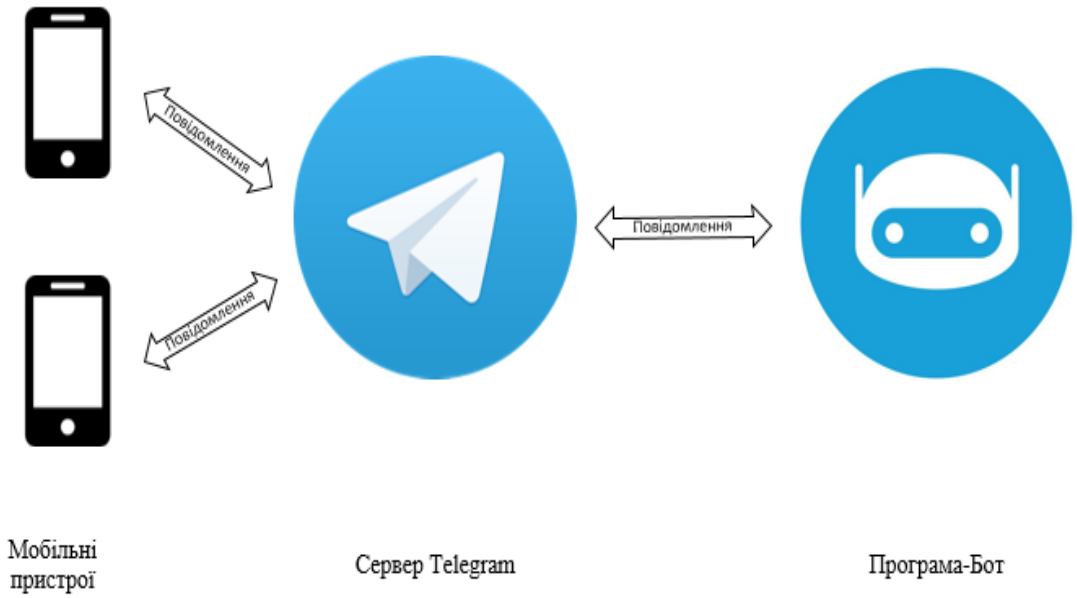


Рисунок 2.1. Архітектура Telegram платформи

Для того, щоб отримувати повідомлення від Telegram використовується два підходи: long polling та webhook. Розглянемо їх детальніше.

2.1.1 Технологія Long Polling

Long Polling – механізм взаємодії у клієнт-серверних системах. Використовуючи її клієнт відправляє запит на сервер, такий самий як і при звичайній взаємодії, але сервер може відповісти не зразу. Якщо на сервері не з’явилася нова інформація з моменту попереднього запиту клієнта то сервер не відповідає порожнім пакетом, він підтримує запит відкритим та чекає доки нова інформація не з’явиться. Як тільки сервер отримає нову інформацію він моментально відправить клієнту відповідь. Клієнт, отримавши відповідь, знову посилає новий запит. Така технологія дозволяє зменшити затримку відповіді (час між моментом появи на сервері нової інформації та моментом відправки клієнтом запиту).

2.1.2 Технологія Webhook

Webhook – механізм отримання повідомлень о деякий подіях між декількома системами. Webhook фактично є колбеком який викликає деяку логіку у відповідь на якусь подію. Для налаштування взаємодії між сервером та клієнтом під логікою розуміється HTTP запит. Коли якась подія виникає на сервері, то сервер посилає HTTP запит на адресу, яку клієнт вказав серверу. Перевагою webhook у порівнянні з long polling є те, що клієнт не надсилає запитів серверу взагалі, та не використовує свої ресурси для цього, він лише чекає доки сервер сам не відправить повідомлення. Для того щоб використовувати webhook у випадку с Telegram клієнт має мати HTTPS сертифікат.

2.2 Огляд алгоритмів машинного навчання

Машинне навчання – клас методів штучного інтелекту особливістю яких є можливість навчатися. Існують два типи навчання: навчання з вчителем, та без вчителя. У обох випадках алгоритм приймає деякі дані для навчання, але у випадку навчання з вчителем дані вже віднесені до певних класів (при задачі класифікації) або для даних задана деяка числова характеристика (для задач регресії). Навчання без вчителя приймає лише дані. Процес навчання полягає у розбитті вибірки на групи (кластери) використовуючи дані о попарній схожості об’єктів вибірки. У даній роботі розглядається спосіб розпізнавання плодових тіл грибів, це є задачею класифікації з багатьма класами. Розглянемо деякі алгоритми які ефективно вирішують цю задачу[4].

2.2.1 Алгоритм kNN

kNN (розшифровується як k Nearest Neighbor або k Найближчих Сусідів) – це один з найпростіших алгоритмів класифікації.

Завдання класифікації в машинному навчанні – це завдання

віднесення об'єкта до одного з заздалегідь визначених класів на підставі його формалізованих ознак. Кожен з об'єктів в цьому завданні представляється у вигляді вектора в N-вимірному просторі. Для навчання класифікатора необхідно мати набір об'єктів, для яких заздалегідь визначені класи.

Для класифікації кожного з об'єктів тестової вибірки необхідно послідовно виконати наступні операції:

- Обчислити відстань до кожного з об'єктів навчальної вибірки;
- Відібрати k об'єктів навчальної вибірки, відстань до яких є мінімальною;
- Клас об'єкта – це клас, який найчастіше трапляється серед k найближчих сусідів.

Перевагами даного алгоритму є те, що реальне навчання не проводиться. Для того, щоб класифікувати об'єкт необхідно лише запам'ятати навчальну вибірку та задати метрику порівняння об'єктів, через це навчання проводиться дуже швидко. Також перевагою є те, що параметр k (кількість сусідів) можна налаштувати шляхом перехресної перевірки.

Недолік цього алгоритму пов'язаний з тим, що якщо сусіди об'єкта відносяться до різних класів рівна і їх кількість рівна, то класифікація буде неможлива (неоднозначна).

Варто зазначити, що у даному досліджені використовується евклідова метрика порівняння об'єктів[11].

2.2.2 Алгоритм Random Forest

Перед тим як розглядати алгоритм Random Forest необхідно описати його складові частини, а саме вирішальні дерева.

Вирішальне дерево або дерево ухвалення рішень — модель, яка використовується у галузі аналізу даних. Вирішальне дерево складається з

двох елементів: «листя» і «гілок». На внутрішніх вершинах («гілках») дерева ухвалення рішення записані предикати від класифікованого об'єкту , в «листі» записані значення класів до яких об'єкти відносяться. Щоб класифікувати новий випадок, треба спуститися по дереву до листа і видати відповідне значення.

Дерева рішень мають такі переваги:

- Простота класифікації;
- Можливість зміни предикатів;
- Можливість оброблювати дані с пропусками;
- Не може виникнути відмова класифікації.

Та такі недоліки:

- Алгоритми побудови дерева жадібні і можуть перенавчатися;
- Велика чутливість до шумів та критерію розбиття вибірки для знаходження предикатів у внутрішніх вершинах дерева.

Тепер можна розглянути алгоритм “Випадковий ліс”.

(Випадковий ліс)— алгоритм машинного навчання що полягає у використанні ансамблю вирішальних дерев.

Алгоритм полягає у наступному:

Нехай навчальна вибірка складається з N об'єктів , розмірність простору ознак дорівнює M, і заданий параметр m

Усі дерева ансамблю будуються незалежно один від одного за такою процедурою:

1. Згенеруємо випадкову підвибірку з повторенням розміром n з навчальної вибірки.

2. Побудуємо вирішальне дерево, яке класифікує приклади даної підвибірки, причому в ході створення чергового вузла дерева будемо вибирати ознаку, на основі якої проводиться розбиття, не з усіх M ознак, а лише з m випадково вибраних.

3. Дерево будується до повного вичерпання підвибірки і не піддається процедурі прунінга (спрощення структури дерева).

Класифікація об'єктів проводиться шляхом голосування: кожне дерево ансамблю відносить об'єкт, який класифікується до одного з класів, і перемагає клас, за який проголосувало найбільше число дерев. У цьому алгоритмі можна налаштовувати такі параметри: кількість вирішальних дерев та параметр m (кількість ознак які випадково обираються для побудови чергового вирішального дерева). Налаштування також проводиться шляхом перехресної перевірки[12].

Переваги алгоритму “Випадковий ліс”:

- Здатність ефективно обробляти дані з великим числом ознак і класів;
- Нечутливість до масштабування значень ознак;
- Однаково добре обробляються як безперервні, так і дискретні ознаки;
- Існують методи побудови дерев за даними з пропущеними значеннями ознак;
- Алгоритм добре виконується паралельно і гарно масштабується, так як всі дерева можуть бути побудовані паралельно.

Недоліки цього алгоритму слідують з недоліків вирішальних дерев, а саме:

- Алгоритм схильний до перенавчання, особливо якщо дані мають велику кількість шумів;
- Великий розмір моделі.

2.3 Огляд мов програмування

У наш час існує багато мов програмування , як спеціальних так і загальних, які можна використовувати для вирішення задач машинного навчання. На рис. 2.2 приведений графік кількості вакансій з різними вимогами до мови програмування. Проаналізувавши цей графік можна зробити декілька висновків. По-перше: у цій сфері використовується досить багато різних мов програмування. По-друге: популярність машинного навчання та науки про дані різко зросла за останні декілька

років. Втретє: Python є лідером, слідом за ним йдуть R, Java та Scala. В останнє: ріст популярності Scala значно зріс. Так 4 роки назад Scala практично не використовувалася для машинного навчання, а зараз вона займає 4 місце. Це може бути викликано зростанням популярності платформи Apache Spark. У даній роботі використовується Python. Порівняємо R, Python та Scala для вирішення задач машинного навчання. Таблиця 2.1 показує кількість запитань на Stack Overflow, кількість пакетів в головному репозиторії та індекс Tiobe (чим менше тим краще). Індекс Tiobe – індекс, що оцінює популярність мов програмування використовуючи дані пошукових запитів, які включають у себе назву мови. Для формування цього індексу використовують такі сервіси, як: Google, проводиться раз у місяць. У випадку Scala треба порахувати кількість Java пакетів, так як Scala, як і Java інтерпритується у байт код та виконується JVM. Порахувати кількість Java пакетів не є об'єктивно можливим.

Таблиця 2.1 – Популярність мов програмування

Запитання на Stack			
Кількість пакетів	packages on PyPL	7,798 packages on CLEAN	Java-пакети
Індекс Tiobe			

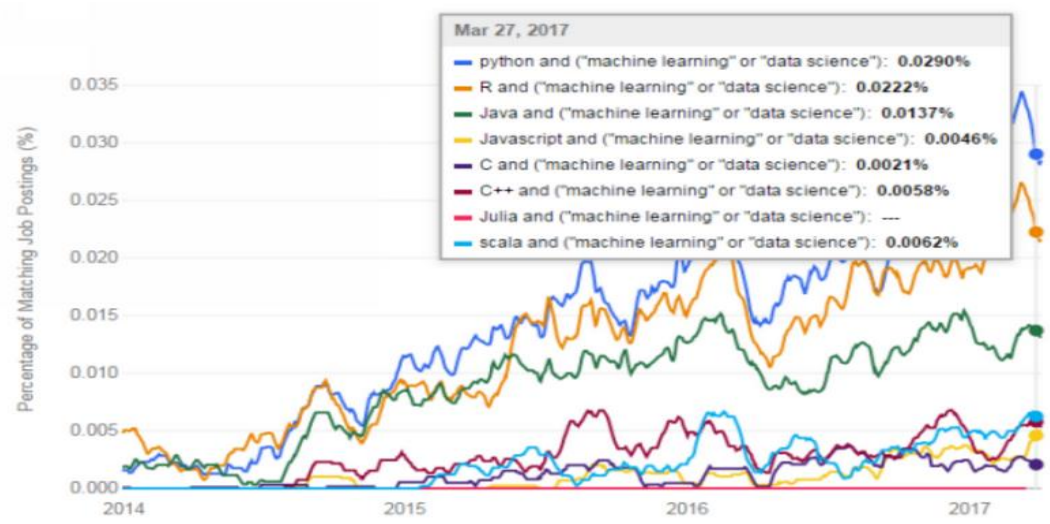


Рисунок 2.2. Графік кількості вакансій для різних мов програмування за 2014 – 2017 роки на посаду аналітика даних

Порівнюючи різні мови треба не тільки акцентувати увагу на швидкості та можливостях, але й на суспільство розробників, які можуть допомагати тим хто починає вивчати нову мову та продовжувати розвиток та ріст різних існуючих мов.

Для розробки комерційних продуктів краще використовувати мови загального призначення, такі як Python, Java та Scala. Спеціальні мови такі як R та Octave може бути кращим в початковій фазі проекту пов'язаного за аналізом даних. Часто у розробці проектів спеціалісти з аналізу даних розроблюють алгоритми на спеціальних мовах, а програмісти потім адаптують їх для використання у комерційному середовищі.

У даній роботі використовується Python. Це пов'язано з тим, що його можна використовувати не тільки для машинного навчання, а й для обробки зображень. Бібліотеки, використані у цей роботі, добре оптимізовані. Фактично всі ресурсозатратні операції виконують код на мові C. Також на мові Python написані фреймворки для розробки ботів для платформи

2.4 Огляд бібліотек машинного навчання

2.4.1 Tensorflow

– бібліотека для побудови нейронних мереж. Вона орієнтована на представлення процесу обрахунку як направленого графу. Це зручно для обчислення великих задач. Tensorflow написаний на мові C++, також ця бібліотека надає можливість викликати ці функції на мові Python, тому швидкість обробки не зменшується. Також Tensorflow підтримує CUDA, програмно-апаратну архітектуру пралаельних обичлень, яка дозволяє збільшити обчислювальну продуктивнчть завдяки використанню графічних процесорів NVidia.

2.4.2 Scikit-learn

Scikit-learn – одна з найпопулярніших бібліотек для машинного навчання. Вона має дуже великий набір інструментів для аналізу даних. Вона побудована використовуючи бібліотеки орієнтовані на наукові дослідження, такі як NumPy, SciPy та matplotlib. Серед усіх бібліотек машинного навчання scikit-learn має найбільш широкий набір алгоритмів та підходів. Вона слугує як фундамент для інших розробок.

2.4.3 Caffe

– бібліотека для вирішення задач візуального розпізнавання. Переважно вона використовується для створення глибоких нейронних мереж. Також ця бібліотека добре зінтегрована з GPU, що дозволяє значно прискорити

обчилення задач, які добре виконуються на графічному процесорі. Caffe найчастіше знаходить застосування у наукових роботах.

2.4.4 Pyevolve

– бібліотека яка використовує генетичні алгоритми. Вона тестує нейрону мережу на деяких даних та використовує генетичні алгоритми для покращення якості моделі.

2.4.5 MLilb

MLlib – бібліотека для машинного навчання, яка використовується платформою Spark, яка використовується для реалізації розподілених систем. Вона орієнтована на масштабованість та паралельність, тому в ній реалізовані ті алгоритми, які можуть виконуватися паралельно. Використовувати MLlib можна на тих мовах, на яких можливо програмувати для платформи Apache Spark, а саме: Python, Scala та Java. MLlib використовує пакет Breeze, написаний на Scala. Він являє собою реалізації алгоритмів машинного навчання на чисельних методів.[9]

2.5 Огляд бібліотек для обробки зображення

Для обробки зображення порівняно 2 популярні бібліотеки: opencv та scikit-image. Так як для обробки зображення використовується scikit-learn , то треба порівняти швидкість операції морфологічного відкриття для того щоб обрати кращу бібліотеку. В таблиці 2.2 приведені результати швидкості обрахування морфологічного відкриття з радіусом структурного елемента рівним 12 для 20 різних зображень. Як видно бібліотека opencv працює швидше у 450 разів, тому саме вона буде використана у роботі. Така велика різниця пояснюється тим, що бібліотека opencv використовує

процедури на мові C, яка компілюється у машинний код, тоді як Python використовує інтерпретатор.

Таблиця 2.2 – Середній час обрахування

		Sckit-mage
Середній час обрахування морфологічного відкриття с радіусом структурного елемента 12, с		

2.6 Огляд фреймворків для розробки Telegram боту

Платформа ботів з’явилася відносно недавно, але програмістам вже доступні фрейморки, які полегшують процес написання ботів. Розглянемо декілька найкращих з багатьох можливих варіантів[12].

2.6.1 Telepot

Telepot – фреймоврк для розробки ботів. Він написаний на мові Python та працює з версіями Python 2.7 та Python 3.5. Цей фреймворк дозволяє використовувати технлогії Webhook та Long Polling для отримання повідомлень, відправляти повідомлення, медіа файли. Також він має підтримку користувацьких клавіатур, та інлайн ботів. Інлайн боти – боти, які викликаються прямо у формі вводу повідомлення, та моментально показують користувачу результати, або змінюють вихідне повідомлення. Перевагами цього фрейморку є його можливість масштабуватися, він може обробляти повідомлення, та відповідати на них паралельно, що є важливим фактором у навантажених системах.

2.6.2 Python-Telegram-Bot

Цей фрейморк написаний на мові Python. Його можна встановити

використовуючи pip, систему керування пакунками. Він, так само як і Telepot оброблює запити паралельно, також він використовує сучасний підхід декораторів у мові Python. Python-Telegram-Bot має широкі можливості до масштабування, може змінювати кількість оброблювачів запитів. Також Python-Telegram-Bot може відновлюватися після збоїв, має розвинуті підходи до логування подій, які трапляються у системі[13].

Висновки до розділу

У даному розділі було виконано огляд відомих підходів до вирішення завдань аналізу. Для класифікації грибів використовують різні алгоритми машинного навчання (ймовірнісна нейронна мережа, метод головних компонент), та різні ознаки (локальні, геометричні ознаки, ознаки кольору, текстури, жилкування). Всі підходи показали гарні результати класифікації. Морфологічний спектр не був застосований у цих роботах, однак він є гарним дескриптором форми, тому на думку автора його використання є перспективним.

Також виконано огляд мов програмування та бібліотек, що використовуються для обробки фото та машинного навчання. Проведено огляд алгоритмів.

Виконано огляд платформи для написання ботів які взаємодіють с месенджером Telegram. До плюсів такого підходу можна віднести те, що цей месенджер можна встановити на будь-яку платформу та отримувати доступ до ботів завжди коли доступно з’єднання з інтернетом.

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Перед тим як приступати до розгляду програмного забезпечення, варто відмітимо деякі особливості морфологічних спектрів. По перше, спектр, що розглядається у даній роботі є стійким до зміни положення об'єкта на зображенні. В першу чергу спектр описує розподіл товщин об'єкта, можна сказати місткість у ньому структурних елементів заданого радіусу. Спектр також відображає іншу структурну інформацію: наприклад, сума значень морфологічного спектра рівна площі об'єкта. Приклад морфологічного спектра у різницевому вигляді приведений на рис. 3.1. Розглянемо основні корисні властивості морфологічного спектра Марагоса.

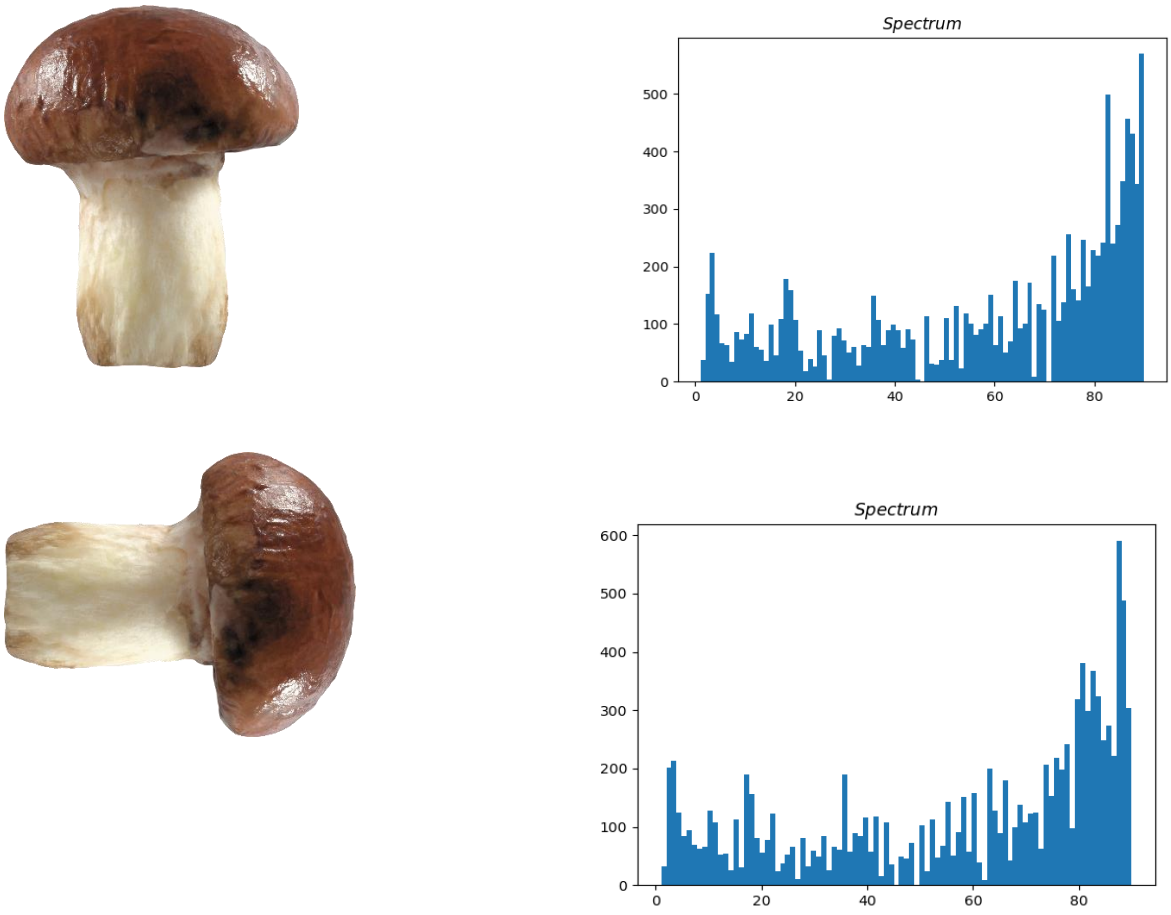


Рисунок 3.1. Перевернуті зображення, які мають приблизно однаковий спектр (похибка виникла тому що зображення подано у дискретному вигляді)

Наявність великих значень в початку спектра означає, що об'єкт має значну кількість малих частин. Наявність великих значень в кінці спектра означає наявність великий частин об'єкту, які вміщують у себе структурний елемент *rB* з великим радіусом *r*.

Залежно від вибору форми структурного елементу спектр може показувати «B-форму» об'єкта, тобто максимальний ступінь входження структурного елемента в об'єкт. На практиці найчастіше використовують структурний елемент у вигляді диска (кола), але існують завдання, коли структурним елементом може виступати квадрат, овал і т.д.

Описані вище особливості дозволяють говорити про актуальність застосування морфологічного спектра в цілому класі задач, пов'язаних з аналізом форми об'єктів, їх класифікацією або кластеризацією. Застосування спектра дозволяє послабити обмеження на положення гриба в кадрі, так як сам спектр не залежить від положення об'єкта.

Таким чином, єдиними обмеженнями в застосовності спектра є різні зміни форми об'єкта в зв'язку з некоректною зйомкою (неправильний ракурс) або в зв'язку з неправильною формою гриба, тому для класифікації можна використовувати фото, зроблені камерою телефону які можна відправити Telegram боту для проведення класифікації.

Також варто додати, що спектр можна представити у різницевому та накопичувальному вигляді. Накопичувальній спектр являє собою різницю між оригінальним зображенням та результатом морфологічного відкриття, а різницевий різницю в площах між результатом від застосування відкриття зі структурним елементом радіуса *r - 1* і радіуса *r*.

3.1 Етапи вирішення задачі класифікації

Приступимо до розгляду задачі. Поділимо її на декілька частин:

- Обробка зображення;
- Побудова морфологічного спектру;

– Обчислення моментів зображення;
– Налаштування алгоритму класифікації.

Схема класифікації показана на рис. 3.2. Розглянемо кожну з цих підзадач.

Рисунок 3.2 Схема класифікації

3.1.1 Обробка зображення

Обробка зображення виконується наступним чином.

На зображенні відбувається пошук контуру найбільшої довжини, після цього відбувається пошук кола з мінімальним радіусом яке описує заданий контур. З зображення вирізається квадрат, що описує знайдене коло. Всі фото приводяться до однакового розміру 400x400 пікселів. Використовуючи такий підхід площа гриба буде приблизно однаковою не залежно від повороту та масштабу (похибка виникає тому що зображення подано у растровому вигляді).

3.1.2 Побудова вектору ознак

Для побудови морфологічного спектру та моментів зображення використано бібліотеку opencv.

3.1.3 Налаштування алгоритму класифікації

Для налаштування класифікатору, так само як і для оцінки якості моделі, користуються перехресною перевіркою. Зазвичай вона використовується в ситуаціях, де метою є класифікація, і хотілося б оцінити, наскільки модель здатна працювати з новими об’єктами, на яких вона не була тренована.

Процедура перехресної перевірки включає:

- Розбиття вибірки на 2 частини: навчальну та контрольну;
- Навчання моделі використовуючи навчальну вибірку;
- Перевірка якості моделі на контрольній вибірці.

Для зменшення дисперсії вибірку випадковим чином розбивають декілька разів і результати декількох перевірок усереднюються.

Перехресна перевірка важлива для захисту від гіпотез, нав'язаних даними, особливо коли отримання додаткових даних важко або неможливо.

Для налаштування параметрів класифікатору його тренують з різними параметрами та для кожної моделі проводять перехресну перевірку. Обирають ті параметри для яких необхідні метрики класифікації кращі.

Також перехресна перевірка допомагає уникати ситуації перенавчання. Розглянемо її детальніше. Припустимо, у нас є модель з одним або декількома невідомими параметрами, і набір даних, на якому ця модель може бути оптимізована (тренувальний набір). Процес тренування оптимізує параметри моделі і робить модель адаптованою під тренувальний набір. Якщо використати дані для перевірки з тренувального набору, зазвичай виявляється, що модель описує тестові дані гірше, ніж тренувальний набір. Це називається перенавчанням (overfitting), і особливо часто зустрічається в ситуаціях, коли розмір тренувального набору невеликий, або коли число параметрів моделі

велике. Перехресна перевірка – це спосіб оцінити здатність моделі працювати на гіпотетичному тестовому наборі, коли такий набір в явному вигляді отримати неможливо.

Розглянемо види перехресної перевірки:

1. Перехресна перевірка по K блокам. В цьому випадку вихідний набір даних розбивається на K блоків. З K блоків один залишається для тестування моделі, а інші K-1 блоків використовуються як тренувальний набір. Процес повторюється K раз, і кожен з блоків використовується один раз як тестовий набір. В результаті виходить K результатів, по одному на кожен блок, вони усереднюються або комбінуються будь-яким іншим способом, та дають одну оцінку. Перевага такого способу перед випадковим семплюванням (random subsampling) в тому, що всі об’єкти вибірки використовуються і для тренування, і для тестування моделі, і кожен об’єкт використовується для тестування в точності один раз. Часто використовується крос-валідація на 10 блоках, але якихось певних рекомендацій щодо вибору числа блоків немає. В пошаровій крос-валідації блоки вибираються таким чином, що середнє значення відповіді моделі приблизно дорівнювало значенням по всіх блоках. На рис. 3.3 зображено пошарове розбиття на блоки на кожній ітерації.

2. Перевірка послідовним випадковим семплюванням (random subsampling). Цей метод випадковим чином розбиває набір даних на тренувальний і тестовий набори. Для кожного такого розбиття, модель навчається на тренувальних даних, а точність роботи моделі оцінюється на тестовому наборі. Результати потім усереднюються по всьому розбитті. Перевага такого методу перед перевіркою на K блоках в тому, що пропорції тренувального і тестового набору не залежать від числа повторень (блоків). Недолік методу в тому, що деякі елементи можуть жодного разу не потрапити в тестовий набір, тоді як інші можуть потрапити в нього більше, ніж один раз.

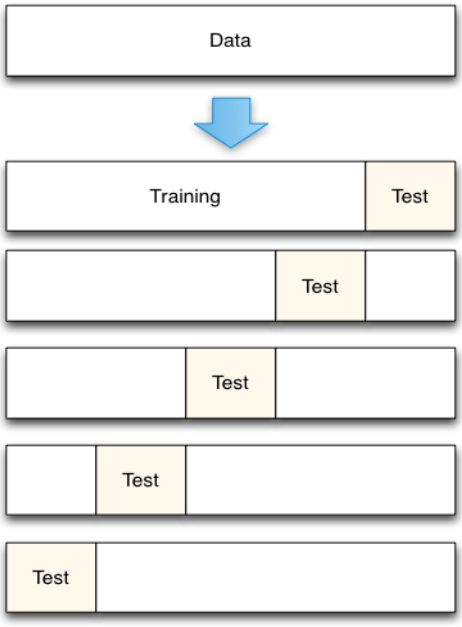


Рисунок 3.3. Розбиття даних на блоки для проведення перехресної перевірки

Іншими словами, тестові набори можуть перекриватися. Крім того, оскільки розбиття проводяться випадково, результати будуть відрізнятися в разі повторного аналізу. У пошаровому варіанті цього методу, випадкові вибірки генеруються так, щоб середня відповідь моделі була однакова на тренувальному і тестовому наборах. Це особливо корисно, коли відповідь моделі з нерівними пропорціями відповідей.

3. Поелементна крос-валідація (Leave-one-out, LOO). Цей метод схожий на перевірку по К блокам де блок складається з одного об’єкту. Цикл перевірки повторюється, поки кожний об’єкт не буде використано один раз в якості тестового.

Мета перехресної перевірки в тому, щоб оцінити очікуваний рівень відповідності якості роботи моделі на нових даних, які є незалежними від даних, на яких модель тренувалася. Вона може використовуватися для оцінки будь-якої кількісної міри якості, що підходить для даних і моделі. Наприклад, для завдання бінарної класифікації, кожен випадок в тестовому наборі буде передбачений правильно або неправильно. У цій ситуації коефіцієнт помилки може бути використаний в якості оцінки відповідності, хоча можуть використовуватися і інші оцінки. Якщо передбачуване значення розподілене неперервно, для оцінки

відповідності може використовуватися середньоквадратична помилка, корінь з середньоквадратичної помилки, або медіанне абсолютне відхилення.

Більшість форм крос-валідації досить прості для реалізації, якщо є готова реалізація методу класифікації. Зокрема, метод класифікації потрібен тільки у вигляді «чорного ящика», немає потреби знати деталі його реалізації. Якщо метод тренування моделі досить ресурсозатратний, крос-валідація може бути повільною, оскільки тренування виконується послідовно багато разів.

3.2 Вибір засобів вирішення задачі

Огляд мов програмування, які ефективно можуть бути використані для вирішення задач класифікації були розглянуті у пункті 2.3. У даній роботі використовується Python. Це пов'язано з тим, що його можна використовувати не тільки для машинного навчання, а й для обробки зображень. Бібліотеки, використані у цей роботі, добре оптимізовані. Фактично всі ресурсозатратні операції виконують код на мові C. Також на мові Python написані фреймворки для розробки ботів для платформи Telegram.

Python має великий набір різних пакетів для наукових робіт, машинного навчання. У даній роботі використана бібліотека scikit-learn. Це пояснюється тим, що у ній реалізовані різні алгоритми машинного навчання, а саме: алгоритми класифікації (kNN, Random Forest) та класеризації (kMeans) який був використаний для обробки зображення. Також scikit-learn має великий набір інструменів для проведення перехресної перевірки, обчислення якості моделі та побудови графіків.

Для написання Telegram боту використано фрейморк Python-Telegram-Bot. Він підходить для використання у даній роботі так як підтримує механізм Long Polling і може приймати зображення від користувачів. Він є найбільш популярним фреймворком на сервісі GitHub та має кращу документацію.

3.3 Архітектура програмного забезпечення

Діаграма класів зображена у додатку 2. Фактично класи, що виконують обробку та класифікацію відповідають елементам рис. 3.2. Розглянемо кожен с класів програми детальніше.

Цей клас використовується для побудови ознак. Для цього були розроблені методи, які обчислюють моменти зображення та морфологічний спектр. Для обчислення моментів зображення та морфологічного відкриття використовується бібліотека opencv.

Цей клас відповідає за обрізку зображення. Він має методи для обробки як кольорових зображень, так і бінарних зображення. Для виділення зображення гриба використовується scikit-learn.

Цей класс використовує класи ImageProcessor та FeatureExtractor для того щоб виконувати класифікацію об'єкта. При створенні екземпляру класу відбувається тренування класифікатору на вилучених із зображень ознак. Після цього можна використовувати екземпляри цього класу для класифікації зображень використовуючи метод `predict(image)`.

Клас Bot використовується для того щоб встановити з'єднання з сервером Telegram та запустити процес отримання фото шляхом механізму Long Polling. Коли бот отримує фото він викликає метод `predict(image)` класу Classifier та відправляє користувачу бота результат класифікації.

3.4 Приклад роботи боту

Перевагою розробки боту для месенджеру Telegram є те, що цей месенджер можна використовувати на будь-якій платфомі. Як на телефоні, планшеті так і на персональному комп'ютері. Telegram можна встановити

на будь-яку платформу Windows, Max OS, Linux. Також Telegram має Web-інтерфейс. Нижче на рис. 3.4 та рис. 3.5 приведені сріншоти роботи з ботом. Для отримання результату необхідно зняти фото гриба на однорідному фоні та відправити його боту.

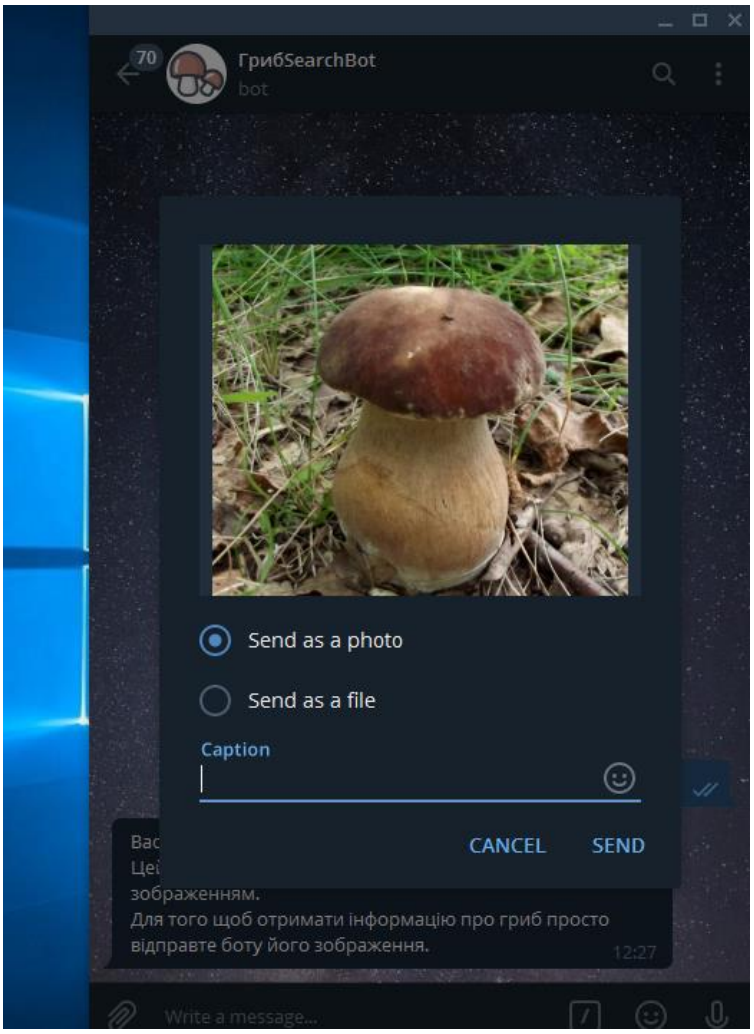


Рисунок 3.4. Завантаження зображення до системи.

Рисунок 3.5. Результат роботи системи

3.5 Тестування роботи алгоритмів

Логічним є питання чому при всіх перевагах спектрального представлення морфологічний спектр не отримав широкого застосування. Це, в першу чергу, пов’язано з високою обчислювальною складністю морфологічних операцій. Фактично, обчислення спектра являє собою багаторазове застосування морфологічного відкриття с примітивами різного радіусу, а операція відкриття практично не може виконуватися паралельно. При цьому час виконання залежить від розміру зображення. В результаті проведених експериментів (таблиця 1) можна зробити висновок, що середній час виконання операції відкриття зростає на 170% при збільшенні зображення на 100 px по кожному з вимірів. При тому, що в середньому для отримання спектру на зображення

400x400 px необхідно зробити 100 операцій відкриття, ріст часу виконання може бути істотним. У таблиці 3.1 наведено порівняння часу виконання операції відкриття с примітивом у вигляді кола з радіусом 25 px.

Таблиця 3.1 – Час виконання операцій

Розмір зображення (px)	Середній час виконання операції відкриття с радіусом примітиву 25 px (сек)
200x200	0.00084
300x300	0.00216
400x400	0.00288
500x500	0.00517

В даній роботі експерименти проводяться з вибіркою зображень розміром 400x400 px. Середній час побудови спектру для такого зображення на комп'ютері с процесором Intel Core i7 7700 3.6 GHz становить 7.85 секунд, що є великими часовими затратами. З такої точки зору використовувати морфологічний спектр для вирішення задач класифікації грибів в режимі реального часу не можливо. Результати експериментального порівняння часу побудови морфологічного спектру приведені у таблиці 3.2.

Таблиця 3.2 – Час побудови спектру

Розмір зображення (px)	Середній час побудови морфологічного спектру
200x200	0.268
300x300	2.478
400x400	7.858
500x500	27.997

3.5.1 Вихідні дані та умови експерименту

Для перевірки застосування морфологічного спектра до вирішення задач класифікації грибів була зібрана вибірка з 130 фото 12 видів грибів які ростуть в Україні. Представлені такі види грибів:

– Печериця польова;
– Мухомор червоний;
– Мухомор пантерний;
– Опеньок справжній;
– Опеньок несправжній;
– Лисичка звичайна;
– Боровик звичайний;
– Бліда поганка;
– Білий гриб;
– Маслюк;
– Маремуха;
– Підберезовик.

Приклад деяких об’єктів зображений на рис. 3.6.

Рисунок 3.6. Елементи вибірки

3.5.2 Результати експерименту

При проведенні експериментів було порівняно два алгоритми класифікації: алгоритм ближнього сусіда (kNN) та випадковий ліс. Для оцінки якості була використана перехресна перевірка. Перехресна перевірка працює наступним чином: фіксується деяка множина розбиття вихідної вибірки на

контрольну та навчальну. Для кожного розбиття виконується налаштування алгоритму по навчальній підвибірці, потім оцінюється середня помилка на об'єктах контрольної вибірки. Оцінкою перехресного контролю є середня по всім розбиттям величина помилки на контрольних об'єктах. В якості об'єктів використовуються морфологічний спектр та моменти зображення.

Алгоритми машинного навчання потребують налаштування задля максимізації якості класифікації. У алгоритмі kNN налаштовують кількість «сусідів», а у алгоритмі Random Forest кількість вирішальних дерев. Також задля підвищення якості класифікації автором був запропонований підхід до зменшення кількості елементів спектру, який полягає у розбитті спектру на частини по n елементів та сумування елементів у них. Параметр n (кількість елементів спектру) також потребує налаштування.

Для визначення кількості сусідів була проведена серія експериментів, яка являла собою перебір значення параметрів і оцінки якості класифікації за допомогою перехресної перевірки. Значення якості обчислювалося як кількість вірно класифікованих об’єктів до всіх об’єктів деякого класу. Результати експериментів для алгоритму kNN приведені на рис. 3.7. та 3.8. Як видно з рисунків алгоритм kNN набагато краще працює зі спектром який поданий у накопичувальній формі. Для обох графіків максимальна точність досягається при використанні одного найближчого сусіда для класифікації об’єкта, це пояснюється компактністю об’єктів у класі. Значення точності досягає 0.32 для різницевого представлення та 0.59 для накопичувального. Такі значення є дуже низькими тому вони потребують подальшого покращення. На рис. 3.7 та 3.8 приведені графіки залежності точності класифікації для алгоритму kNN від параметру n що характеризує розмір частини, на які розбивається спектр. Кількість сусідів у обох експериментах дорівнювала одному. Для різницевого спектру максимальне значення класифікації дорівнює 0.67 і воно досягається при n що дорівнює 26. Для накопичувального спектру ці значення дорівнюють 0.738 та 26 відповідно.

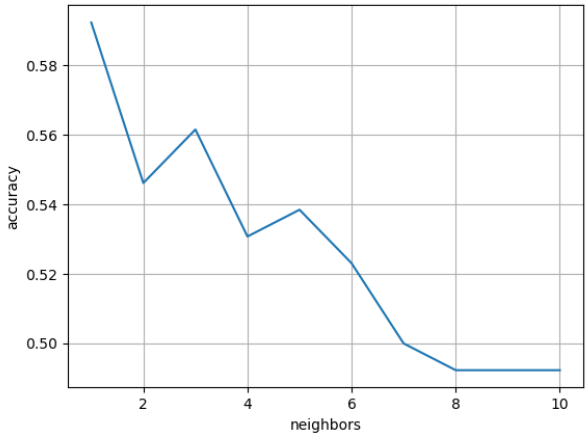


Рисунок 3.7. Залежність точності алгоритму класифікації грибів від кількості сусідів для накопичувального типу спектру

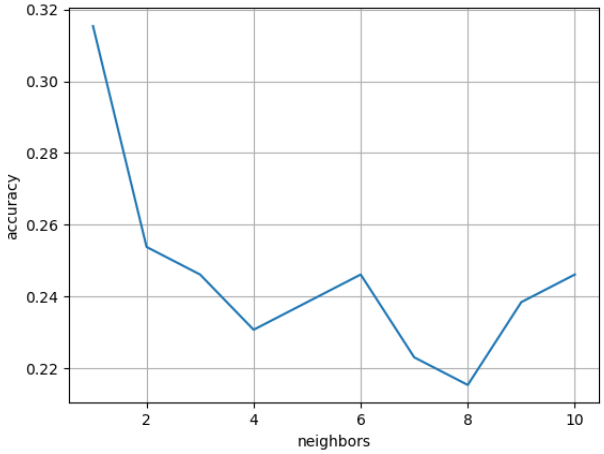


Рисунок 3.8. Залежність точності алгоритму класифікації грибів від кількості сусідів для різницевого типу спектру

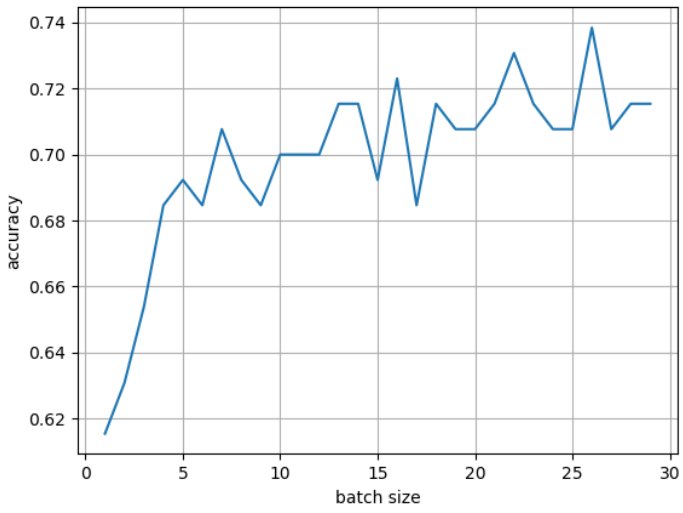


Рисунок 3.9. Залежність точності алгоритму класифікації грибів від розміру розбиття для накопичувального типу спектру

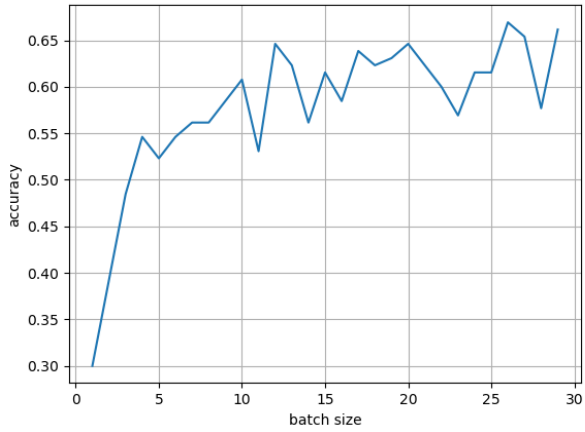


Рисунок 3.10. Залежність точності алгоритму класифікації грибів від розміру розбиття для різницевого типу спектру

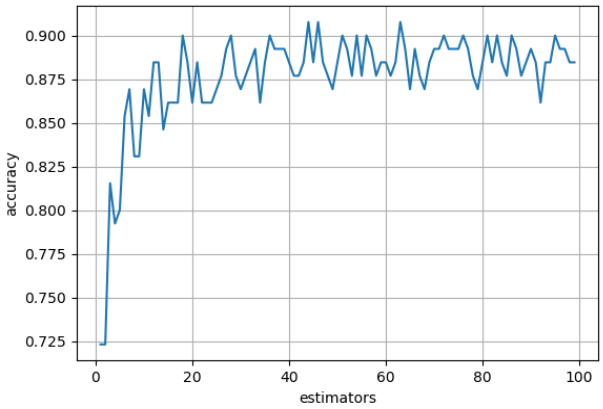


Рисунок 3.11. Залежність точності алгоритму класифікації грибів від кількості вирішальних дерев для накопичувального типу спектру

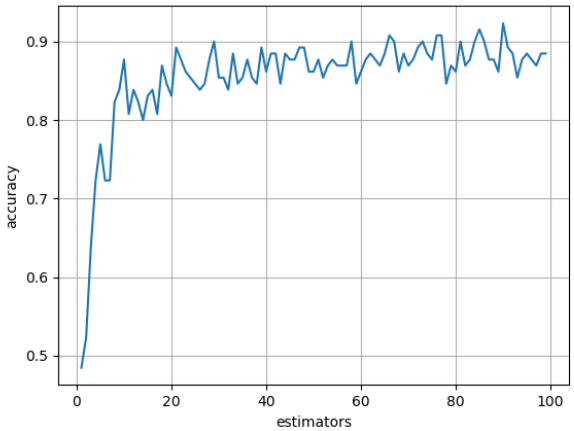


Рисунок 3.12. Залежність точності алгоритму класифікації грибів від кількості вирішальних дерев для різницевого типу спектру

Для визначення кількості вирішальних дерев в алгоритмі Random Forest була проведена така ж серія експериментів як і для kNN. Результати експериментів для алгоритму kNN приведені на рис. 3.11 та 3.12. Як видно з рис. 3.13 та 3.14 алгоритм Random Forest має приблизно однакові показники якості класифікації. Значення точності досягає 0.923 для різницевого представлення при кількості дерев рівній 90 та 0.907 для накопичувального при 44 вирішальних деревах.

Так як алгоритм Random Forest показав набагато кращий результат класифікації в порівнянні з алгоритмом kNN, то він буде використовуватися як класифікатор в програмі класифікації грибів.

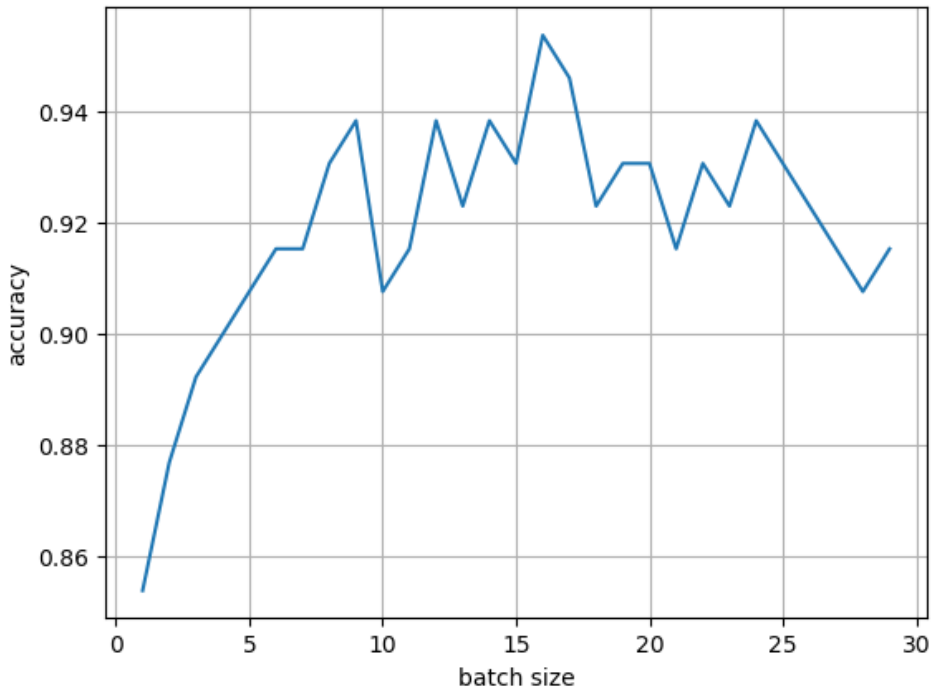


Рисунок 3.13. Залежність точності алгоритму класифікації грибів від розміру розбиття (алгоритм Random Forest) для накопичувального типу спектру

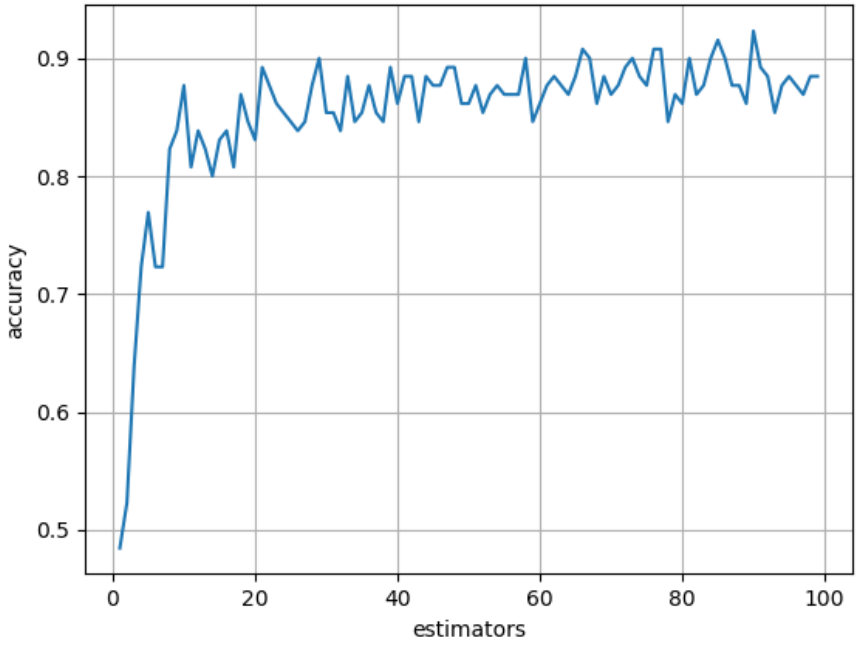


Рисунок 3.14. Залежність точності алгоритму класифікації грибів від розміру розбиття (алгоритм Random Forest) для різницевого типу спектру

Висновки до розділу

У цьому розділі виконано огляд етапів вирішення задачі класифікації. Проаналізовано властивості морфологічного спектру, які доводять те, що для класифікації можна використовувати фото, зроблене камерою телефону, що дає можливість використовувати платформу ботів для класифікації зображень.

Проаналізовано та обрано інструменти та бібліотеки для реалізації програми. Проведено огляд класів, які представлені у програмному продукті та продемонстровано інтерфейс користувача при використанні розробленого боту.

Python обрана як мова програмування. Це пояснюється тим, що Python активно використовується для вирішення задач машинного навчання. На ньому написані необхідні бібліотеки швидкість яких обумовлена тим, що на мові Python можна викликати процедури C.

У даному розділі розглянуто вихідні дані умови експерименту, проведено огляд алгоритмів, які використовуються у даному дослідженні та проведені експерименти.

Приведені види дерев які є входять у навчальну вибірку та можуть бути класифікованими.

Алгоритми натреновані використовуючи спектри, що подані у двох виглядах: різницевому на накопичувальному. Для кожного набору ознак та алгоритмів були проведені експерименти для підбору параметрів моделі. Найкращий результат показав алгоритм «Випадковий ліс» використовуючи спектр у накопичувальному вигляді з кількістю вирішальних дерев рівною 90 та розміром розбиття спектру рівним 15. Міра точності роботи алгоритму була обрахована використовуючи перехресну перевірку. Алгоритм показав 85% точність класифікації.

РОЗДІЛ 4. МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї проекту (товару, послуги, технології)

Основною метою роботи є створення інтелектуальної системи, на основі нейронної мережі для розпізнавання видів грибів, а також розробка Telegram-бота, який надає можливість користувачу класифікувати гриби по фото, завдяки взаємодії зі створеною системою.

Таблиця 4.1 — Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розпізнавання видів грибів	1. Класифікація грибів.	Зменшення можливості отруїтись, економія часу.
	2. Отримання короткої інформації щодо гриба.	Можливість отримати нові знання щодо знайденого гриба.
	3. Наукова діяльність.	Застосування новітніх технологій.

Таблиця 4.2 — Порівняння техніко-економічних характеристик ідеї з потенціальними конкурентами.

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			
		Мій проект	Гриби: велика енциклопедія	Розпізнавання грибів по фото	Довідник грибника
1.	Вартість ПЗ	Низька	Висока	Середня	Висока
2.	Час обробки	Низький	Високий	Низький	Низький
3.	Точність	Висока	Низька	Низька	Середня

.	Споживачі (відомий бренд)	-	+	+	+

Таблиця 4.3 — Визначення сильних, слабких та нейтральних характеристик ідеї проекту.

№ п/п	Техніко-економічні характеристики ідеї	W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
1.	Вартість			+
2.	Точність		+	
3.	Швидкодія		+	
4.	Доступність			+
5.	Використання інтернету	+		

Дана система є «розумною», хоч і має аналоги по своєму функціоналу, проте вони мають ряд недоліків. Крім цього, система є відносно недорогою, що робить програмне забезпечення конкурентоспроможним і перспективним.

4.2 Технологічний аудит ідеї проекту

Таблиця 4.4 — Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Розпізнавання видів грибів	Нейронна мережа	Наявні.	Так.
2.	Надання корисної інформації	Telegram-бот	Наявні.	Так.

Висновок: технологічна реалізація проекту доступна.

4.3 Попередня характеристика потенційного ринку стартап-проекту

Головною перевагою та характеристикою даного продукту є співвідношення ціни та можливостей, що відповідають високому рівню якості. Зависока ціна існуючих аналогів та їх недоступність для широкого кола споживачів робить даний товар перспективним для застосування як в Україні, так і закордоном.

Таблиця 4.5 — Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1.	Кількість головних гравців, од.	3
2.	Загальний обсяг продаж, ум.од/час	1000
3.	Динаміка ринку (якісна оцінка)	Зростає
4.	Наявність обмежень для входу (вказати характер обмежень).	Новітня технологія потребує ресурсів, практичної перевірки та доробки універсального алгоритму.
5.	Специфічні вимоги до стандартизації та сертифікації.	Не потребує сертифікації.
6.	Середня норма рентабельності в галузі (по ринку), %	50

Висновок: враховуючи кількість головних гравців по ринку, зростаючу динаміку ринку, невелику кількість конкурентів та середню норму рентабельності можна зробити висновок, що на даний момент, ринок для входження стартап-продукту є привабливим.

4.4 Визначення груп потенційних клієнтів.

Основною споживчою аудиторією є усі люди, які займаються збором та споживанням грибів та переживають за своє життя та здоров'я. Розповсюдження інтелектуальної системи відбувається через Telegram.

Таблиця 4.6 — Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів
1. Дешевизна технологій		Зацікавлені в купівлі дешевшого.	Задовільна цінова політика, відповідність ДСТУ/ ISO, сумісність з ПК, висока якість, гарні технічні характеристики (час обробки, точність).
2.Зменшення ризику виникнення людського фактору.	Зручний інструмент для розпізнавання видів грибів	Зацікавлені в кращій якості.	
3. Купувати вітчизняне		Зацікавлені в підтримці національного виробника.	
4.Отримати закордонний аналог по якості за українською ціною		Зацікавлені за менші гроші отримати високоякісний матеріал.	

Висновок: визначена характеристика дозволяє зробити висновок, що товар знайде попит у потенційних клієнтів.

4.5 Аналіз ринкового середовища.

Для досягнення необхідного рівня конкурентоспроможності, що являється основною загрозою для інших виробників такої продукції, потрібна гідна реклама.

Таблиця 4.7 Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Конкурентоспроможність	Новий товар на ринку– невідомий бренд, низька конкурентоспроможність	Реклама, вдалий маркетинговий проект, залучення спонсорів до співпраці.
2.	Невідома система	Невідомий новий товар невідомої фірми.	Розкрутка товару.
3.	Комерційна таємниця	Можливість відкриття алгоритму невідомими робітниками.	Підписання строгих контрактів з несенням матеріальної компенсації.

Таблиця 4.8 — Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Новий продукт	Вихід на ринок, зменшення монополії, надання нових рішень у даній сфері.	Вихід нової продукції на ринок, розробка нової функціональності.
2.	Вихід аналогу	Надати продукт з характеристика та можливостями, що відсутні у конкурентів.	Аналіз ринку та користувачів задля задоволення їх потреб та надання функціональності у найкоротші строки за ціну, котра є дешевшою ніж у продуктів-замінників.
3.	Зворотній зв’язок від користувачів	Можливість отримання необхідної інформації для вдосконалення продукту	Наявність вхідних даних та реакція на них з боку команди розробників задля задоволення потреб та бажань кінцевих користувачів.
4.	Грошова винагорода за рекламу	При достатньому попиту на систему розпізнавання видів грибів можлива комерціалізація продукту на основі реклами задля отримання грошової винагороди для подальшого розвиту продукту та оплати заробітної плати працівникам	Точкова комерціалізація продукту; Введення реклами; Ведення додаткових коштів у проект задля його подальшого розвитку.

Висновок: стартап-проект можна впроваджувати на ринок.

4.6 Аналіз пропозиції

Таблиця 4.9 — Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції	Олігополія	Розкрутка національного товару, залучення інвесторів.
2. За рівнем конкурентної боротьби	Всі продукти замінники розроблялись інтернаціональними командами з різних куточків світу, продукти не належать до певної держави, а належать команді розробників	Вихід на ринок збуту продукту з клієнто-необхідною функціональністю, налагодження маркетингу на основних Інтернет ресурсах задля охоплення великої кількості потенційних користувачів
3. За галузевою ознакою	Внутрішньогалузева	Використовується для розпізнавання видів грибів
4. Конкуренція за видами товарів	Товарно-родова	Власні розробки, унікальна технологія, реклама.
5. За характером конкурентних переваг	Цінова	Ціна нижча за аналоги аналоги.
6. За інтенсивністю – марочна/не марочна	Не марочна	Реклама, покращення технологій

Проаналізувавши можливості роботи на ринку з огляду на конкурентну ситуацію можна зробити висновок: оскільки кожний з існуючих продуктів не впливає у великій мірі на поточну ситуацію на ринку в цілому, кожний з існуючих продуктів має свою специфічну сферу використання та свої позитивні

та негативні сторони щодо рішення певних типів задач, то робота та вихід на даний ринок є можливою і реалізованою задачею.

4.7 Аналіз конкуренції у галузі за Майклом Портером

Таблиця 4.10 — Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Існують	Гнучкі ціни, розмір капіталовкладень	Велика концентрація постачальників	Якісна продукція, не потребує великої кількості товару	Вища ціна
Висновки	Доволі висока конкурентна боротьба	Є можливості входу в ринок, наявні потенційні конкуренти	Постачальники диктують умови роботи на ринку, наприклад, ціну та швидкість розповсюдження	Так, залежно від попиту на товар.	Обмежень немає

4.8 Перелік факторів конкуренто-спроможності

Таблиця 4.11 — Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Час	Економія часу користувача.
2	Ціна	Ґрунтується на собівартості – отже є нижчою.

3	Якість	Не гірша за існуючі аналоги.

Висновок: наведені фактори дозволяють вважати бізнес-продукт конкуренто-спроможним та продукт може бути успішно реалізованим на ринку.

4.9 Аналіз сильних та слабких сторін стартап-проекту з урахуванням визначених факторів конкуренто-спроможності

Таблиця 4.12 — Порівняльний аналіз сильних та слабких сторін власного проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з власною компанією						
			–3	–2	–1	0	+1	+2	+3
1	Час	15		+					
2	Ціна	18	+						
3	Якість	18	+						

Висновок: рейтинг власного проекту вище, зважаючи на ціновий фактор та час оброки.

4.10 Аналіз можливості впровадження стартап-проекту на ринок

Таблиця 4.13 — SWOT-аналіз стартап-проекту

Сильні сторони: Собівартість Ціна Рентабельність	Слабкі сторони: Невідомий бренд
Можливості: Вихід на закордонний ринок Забезпечення споживчих потреб Дохід	Загрози: Викриття комерційної таємниці Недостатня реалізація

Висновок: проведений SWOT-аналіз показав, що стартап-проект доцільно реалізовувати у ринкових умовах нашої країни.

4.11 Перелік заходів для виведення стартап-проекту на ринок

Таблиця 4.14 — Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Практичне використання алгоритму, вдосконалення	+	1 рік
2	Залучення іноземних фахівців	-	1 рік
3	Рекламна кампанія	+	1,5 року
4	Презентація товару на хакатонах й інших ІТ заходах	Ресурс – час та гроші для участі, наявні	1-3 місяці
5	Вихід на закордонний ринок	+	6 років

Обраною альтернативою є залучення рекламної кампанії, розкрутка нової системи.

Висновок: вище зазначені заходи дозволять вивести стартап-проект на ринок.

4.12 Розроблення ринкової стратегії проекту

Таблиця 4.15 — Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент

1	Люди, які вживають гриби	Готові	Дуже необхідно	Середня	Легка
2	Дослідники	Готові	Необхідно	Середня	Легка
Які цільові групи обрано: усі споживачі, що бережуть своє життя та здоров'я					

Відповідно до проведеного аналізу можна зробити висновок, що підходящою цільовою групою для розповсюдження даного програмного продукту є споживачі грибів. Відповідно до стратегії охоплення ринку збуту товару обрано стратегію масового маркетингу.

Таблиця 4.16 — Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Споживачі грибів	Стратегія лідерства по витратах	– ціна – простота застосування – економія часу – продуктивність – низькі витрати	Зменшення часу на класифікацію грибів

Таблиця 4.17 — Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки

1	Ні	Буде шукати	Ні, продукт застосовує власний алгоритм для ПЗ.	Стратегія наслідування лідера

Таблиця 4.18 — Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Дешевизна закупівлі.	Стратегія наслідування лідера	– ціна – простота виготовлення – кількість виробленої продукції – якість	– зниження цін – вітчизняне виробництво – невеликі обсяги – універсальність для кожного клієнта
2	Зменшення ризику людської помилки.			
3	Купувати вітчизняне.			
4	Отримати якісний продукт за приємною ціною.			

Висновок: обрані стратегії сприятимуть швидкому виходу на ринок.

4.13 Розроблення маркетингової програми стартап-проекту

Таблиця 4.19 — Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Класифікація	Точність та швидкість	Висока точність розпізнавання
2	Інформація	Надання інформації	Інформація постійно оновлюється

Таблиця 4.20 — Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
1.Товар за задумом	Програмне забезпечення надає змогу припаркувати автомобіль без участі водія.		
2.Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Висока швидкість 2. Точність 3. Кросплатформенність		
	Якість: стандарти, нормативи, параметри тестування тощо Стандартизація відповідно до ДСТУ,ISO. Регламентується НД, СРМ.		
	Пакування відсутнє.		
	Марка: назва організації – розробника + назва товару		
Потенційний товар буде захищено від копіювання: патентування, сертифікати відповідності.			

Таблиця 4.21 — Визначення меж встановлення ціни (із розрахунку на 1 рік користування)

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
>100	150-200грн.	25000 грн	70-400 грн

Таблиця 4.22 — Формування системи збуту

п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
	Користувачі можуть самостійно підключити бота та отримати інструкцію щодо користування ним	Маркетингові дослідження, обслуговування проданих товарів, Прийняття на себе ризику торгових угод	Канал нульового рівня (прямий маркетинг)	Telegram

Таблиця 4.23 — Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1.	Довіра до технологічних інженерів, що рекомендують товар	Канали та пабліки в Telegram	Міжособистісні комунакації	Донести основну ідею, цінову політику, якість	Продемонструвати переваги перед існуючим товарами

Як результат створено ринкову (маркетингову) програму, що включає в себе визначення ключових переваг концепції потенційного товару, опис моделі товару, визначення меж встановлення ціни, формування системи збуту та концепцію маркетингових комунікацій.

Висновки до розділу

У четвертому розділі описано стратегії та підходи з розроблення стартап-проекту, визначено наявність попиту, динаміку та рентабельність

роботи ринку, як висновок вказано що існує можливість ринкової комерціалізації проекту. Розглянувши потенційні групи клієнтів, бар'єри входження, стан конкуренції та конкурентоспроможність проекту було встановлено що проект є перспективним. Розглянуто та вибрано альтернативу впровадження стартап-проекту та доведено доцільність подальшої імплементації проекту. Бар'єри входження на ринок відносно невисокі, через наявну конкуренцію, але завдяки низькій вартості розробленої інтелектуальної системи та її встановлення, існує можливість залучити велику кількість користувачів. Тому, конкурентоспроможність проекту достатньо висока.

ВИСНОВКИ

В даній роботі був розглянутий один з підходів до вирішення задачі класифікації об’єктів за даними зображення. Метод, який базується на побудові спектрів, тобто дескрипторів фігури, не має суттєвих обмежень до об’єктів вибірки та обчислювальним ресурсам, при цьому він має багато корисних властивостей, наприклад регулярність, незалежність до положення об’єкта у кадрі, масштабу. Цей підхід може використовуватися для вирішення задач, аналогічних досліджуваної в даній роботі.

В дисертації було реалізовано інтелектуальну систему що дозволяє визначати вид грибів за даними зображення та бот месенджеру Telegram, який дає змогу зручно взаємодіяти користувачеві з системою та забезпечує можливість користуватися системою використовуючи велику кількість пристроїв, від персональних ком’ютерів до мобільних пристроїв не залежно від операційної системи.

Результати роботи системи можуть значно поліпшити життя людини, запобігти отруєнням та завадити виникненню летальних випадків.

В дипломній роботі експериментально показана ефективність застосування описання об’єктів за допомогою спектрів та моментів зображення. Результат оцінки якості алгоритму показав точність класифікації у 85%.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Українські національні новини. Статистика грибних отруєнь в Україні. Київ, 2018. URL: https://www.unn.com.ua/uk/news/1753741-statistika-gribnikh-otruyen-v-ukrayini-u-moz-nazvali-kilkist-zhertv.

2. Шапинкові гриби, будова та значення, будова клітини гіфи, характер живлення та розмноження, правила збирання грибів. Київ, 2015. URL: http://8next.com/botan/5735-botan_708.html.

3. Шапинкові гриби. Київ, 2016. URL: https://moyaosvita.com.ua/ bibiologija/shapinkovi-gri/.

4. Bhadeshia H. K. D. H. Neural Networks in Materials Science. ISIJ International, 1999. №34 P: 966–979. DOI:10.2355/isijinternational.39.966.

5. Siegelmann H.T., Sontag, Eduardo D. Analog computation via neural networks. Theoretical Computer Science, 1994. №131 P: 331–360. DOI:10.1016/0304-3975(94)90178-3.

6. Рыбина Г. В. Основи построения интелектуальных систем. Финансы и статистика. ИНФРА-М, 2010. с. 432.

7. Смолин Д. В. Введение в искусственный интеллект: конспект лекций. ФИЗМАТЛИТ, 2003. с. 208.

8. M. K. Hu, "Visual Pattern Recognition by Moment Invariants", IRE Trans. Info. Theory, vol. IT-8, pp.179–187, 1962

9. Petros Maragos. Pattern Spectrum and Multiscale shape representation / Petros Maragos., 1987.

10. Визильтер юрий валентинович. Обобщенная проективная морфология / Визильтер Юрий Валентинович. // Институт систем обработки изображений Российской академии наук. – 2008. – №4.

11. A Feature Extraction Method Based on the Pattern Spectrum for Hand Shape Biometry / Juan Manuel Ramirez-Cortes,, Pilar Gomez-Gil,, Gabriel Sanchez-Perez,, David Baez-Lopez., 2008. – (IA ENG).

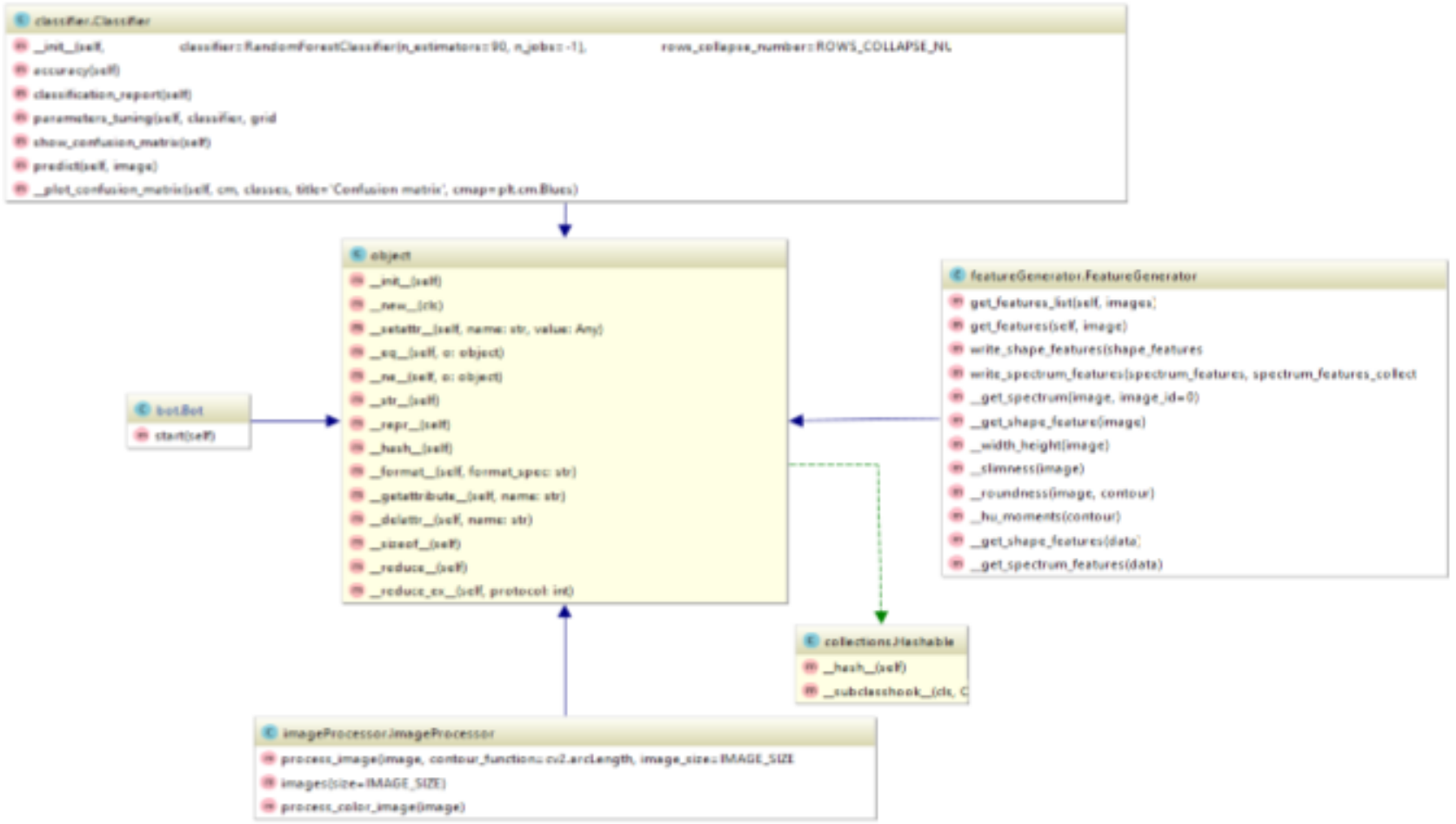
12. Egmont-Petersen M., de Ridder D., Handels H. (2002). Image processing with neural networks – a review. Pattern Recognition, 2002. №39 P: 2279–2301. DOI:10.1016/S0031-3203(01)00178-9.

13. Zhihu Huang. Analysis of Hu's Moment Invariantson Image Scaling and Rotation / Zhihu Huang, Jinsong Leng. – Edith C owan University, 2010.

ДОДАТКИ

ДОДАТОК А
Схема взаємодії системи

Схема взаємодії системи

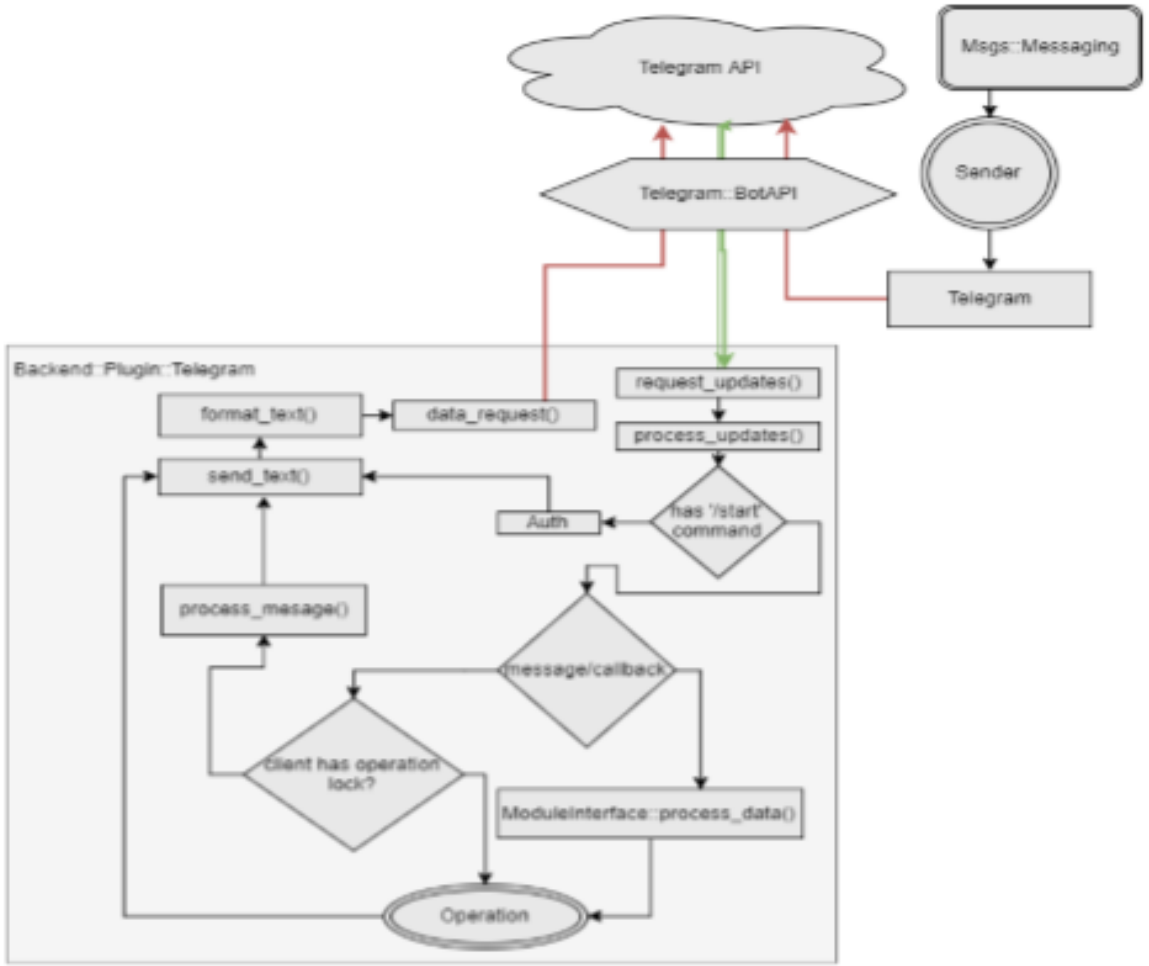


Демонстраційний плакат № 1
до дипломної роботи на тему
„Інтелектуальна система розпізнавання видів грибів"

Розробив: _____
Прийняв: _____

ДОДАТОК Б
Схема роботи Telegram бота

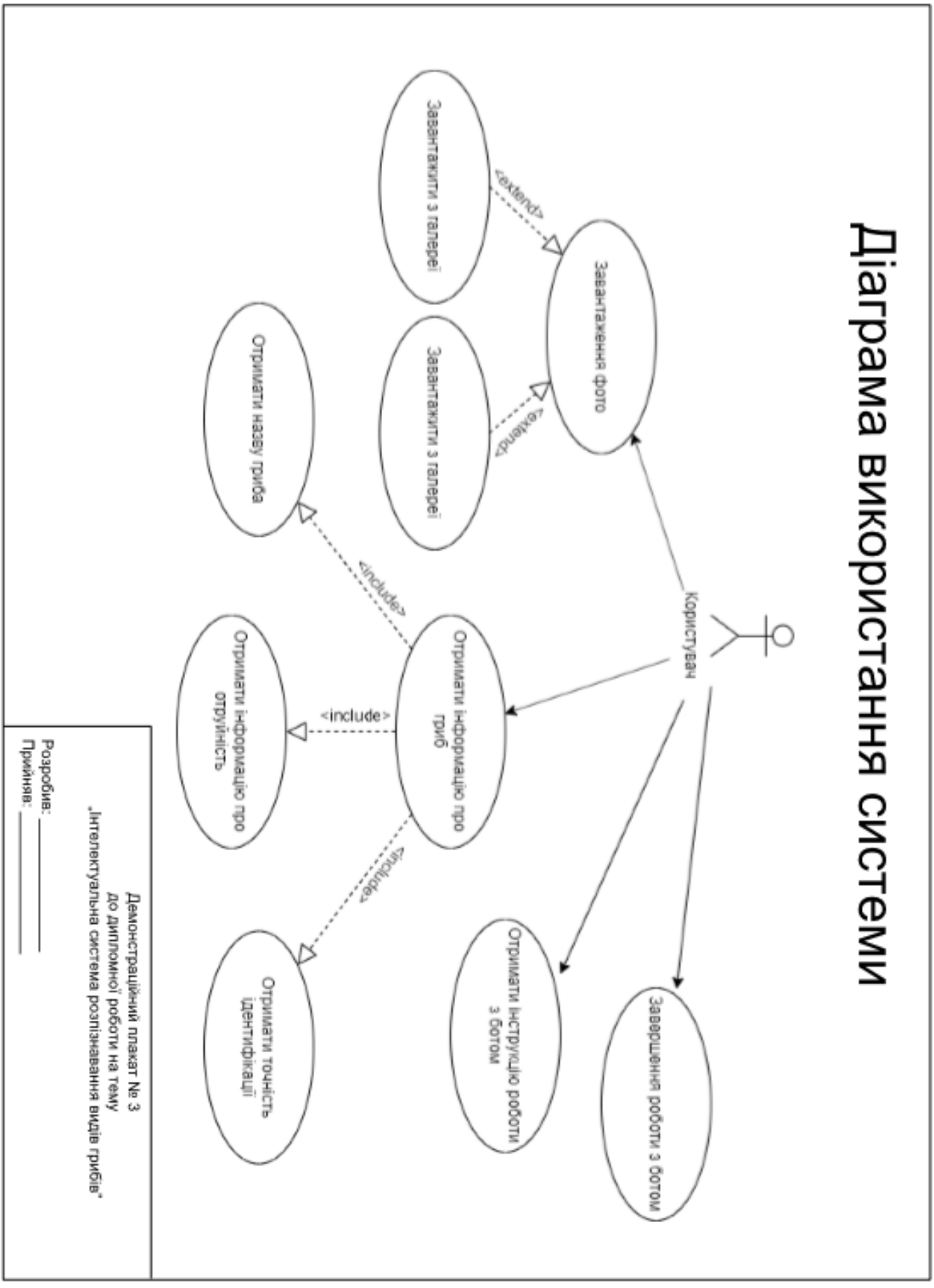
Схема роботи Telegram бота



Демонстраційний плакат № 2
до дипломної роботи на тему
„Інтелектуальна система розпізнавання видів грибів"

Розробив: _____
Прийняв: _____

ДОДАТОК В
Діаграма використання системи


Діаграма використання системи
Користувач
Завантаження фото
Завантажити з галереї
Завантажити з галереї
Отримати назву гриба
Отримати інформацію про гриб
Отримати інформацію про отруйність
Отримати точність ідентифікації
Отримати інструкцію роботи з ботом
Завершення роботи з ботом
<extend>
<extend>
<include>
<include>
<include>
Демонстраційний плакат № 3
до дипломної роботи на тему
„Інтелектуальна система розпізнавання видів грибів"
Розробив: _____
Прийняв: _____

ДОДАТОК Г
Результат перевірки на співпадіння

ДОДАТОК Д
Лістинг коду програми

```
   class FeatureGenerator:
       @staticmethod
       def generate():
           processed = ImageProcessor.images()
           shape_features                                          =
FeatureGenerator.__get_shape_features(processed)
           spectrum_features,       spectrum_features_collect      =
FeatureGenerator.__get_spectrum_features(processed)
           FeatureGenerator.__write_shape_features(shape_features)

FeatureGenerator.__write_spectrum_features(spectrum_features,
spectrum_features_collect)

       @staticmethod
       def get_features(image):
           spectrum,                   spectrum_c                  =
FeatureGenerator.get_spectrum(image)
           shape = FeatureGenerator.get_shape_feature(image)
           return spectrum, spectrum_c, shape

       @staticmethod
       def get_spectrum(image, image_id=0):
           """Returns morphological spectrum in two forms
               :arg image np.array, grayscaled (base 255) image"""
           threshold(image)
           image = image // 255
           original_area = np.array(image).sum()
           i = 3
           spectrum = []
           spectrum_increase = []
           opening_area = original_area
           previous_area = original_area
           t = current_milli_time()
           print(threading.current_thread().getName(), image_id)
           while opening_area > 0:
               kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,
(i, i))
               opening  =  cv2.morphologyEx(image,  cv2.MORPH_OPEN,
kernel)
               opening_area = opening.sum()
               spectrum_increase.append(original_area              -
opening_area)
               spectrum.append(int(previous_area)                  -
int(opening_area))
               previous_area = opening_area
               i += 2
           print(threading.current_thread().getName(),     image_id,
current_milli_time() - t)
           return spectrum, spectrum_increase

       @staticmethod
```

```
        def get_shape_feature(image):
            area = image.sum()
            assert (area % 255 == 0)
            area /= 255
            contour = max_contour(np.array(image), cv2.arcLength)
            image = image // 255
            s = FeatureGenerator.__slimness(np.array(image))
            r     =     FeatureGenerator.__roundness(np.array(image),
contour)
            w, h = FeatureGenerator.__width_height(np.array(image))
            hu = FeatureGenerator.__hu_moments(contour)
            return [s, r, w, h, area] + list(hu)

        @staticmethod
        def __width_height(image):
            x, y, w, h = cv2.boundingRect(image)
            if w > h:
                w, h = h, w
            return w, h

        @staticmethod
        def __slimness(image):
            x, y, w, h = cv2.boundingRect(image)
            if w > h:
                w, h = h, w
            return w / h

        @staticmethod
        def __roundness(image, contour):
            perimeter = contour
            perimeter = cv2.arcLength(perimeter, True)
            imageArea = image.sum()
            return 4 * math.pi * imageArea / (perimeter * perimeter)

        @staticmethod
        def __hu_moments(contour):
            M = cv2.moments(contour)
            return cv2.HuMoments(M).flatten()

        @staticmethod
        def __get_shape_features(data):
            features = []
            for id, label, image in data:
                features.append([label]                                 +
FeatureGenerator.get_shape_feature(image))
            features.sort(key=lambda item: item[0])

            return features

        @staticmethod
        def __get_spectrum_features(data):
```

```
            def thread(f, f_inc, data):
                for id, label, image in data:
                    spectrum,           spectrum_increase           =
FeatureGenerator.get_spectrum(image, id)
                    f.append([label] + spectrum)
                    f_inc.append([label] + spectrum_increase)

            threads = []
            features = []
            features_inc = []
            data = split(data, THREADS_NUM)
            for df in data:
                f = []
                ff = []
                features.append(f)
                features_inc.append(ff)
                threads.append(threading.Thread(target=thread,
args=(f, ff, df)))

            for t in threads:
                t.start()

            for t in threads:
                t.join()

            features = list(itertools.chain(*features))
            features_inc = list(itertools.chain(*features_inc))

            features.sort(key=lambda item: item[0])
            features_inc.sort(key=lambda item: item[0])

            max_spectrum = 0

            for feature in features:
                max_spectrum = max(len(feature), max_spectrum)

            for i in range(len(features)):
                features[i] = features[i] + [0] * (max_spectrum -
len(features[i]) + 1)

            max_spectrum = 0

            for feature in features_inc:
                max_spectrum = max(len(feature), max_spectrum)

            for i in range(len(features_inc)):
                a = features_inc[i][-1]
                features_inc[i]   =   features_inc[i]   +   [a]   *
(max_spectrum - len(features_inc[i]) + 1)

            return features, features_inc
```

```
      @staticmethod
      def __write_shape_features(shape_features):
          with open(SHAPE_FEATURES_FILE_PATH + '--' + time_string(),
'w', newline='') as csvfile:
              fieldnames = ['label', 'slimness', 'roundness',
'width', 'height', 'area',
                            'hu1', 'hu2', 'hu3', 'hu4', 'hu5',
'hu6', 'hu7']
              writer = csv.writer(csvfile)
              writer.writerow(fieldnames)
              for feature in shape_features:
                  writer.writerow(feature)

      @staticmethod
      def              __write_spectrum_features(spectrum_features,
spectrum_features_collect):
          def write(name, data):
              l = len(data[0])
              with open(name, 'w', newline='') as csvfile:
                  fieldnames = ['label'] + list(map(str, range(1, l
+ 1)))
                  writer = csv.writer(csvfile)
                  writer.writerow(fieldnames)
                  for feature in data:
                      writer.writerow(feature)

          write(SPECTRUM_FEATURES_FILE_PATH + '--' + time_string(),
spectrum_features)
          write(SPECTRUM_FEATURES_COLLECT_FILE_PATH + '--' +
time_string(), spectrum_features_collect)



  class ImageProcessor:

      @staticmethod
      def process_image(image, contour_function=cv2.arcLength,
image_size=IMAGE_SIZE):
          """Returns cropped images with fixed size.
              @:arg image - np.array
          """
          (rows, cols) = image.shape
          i = np.array(image)
          cnt = max_contour(image, contour_function)
          (x, y), radius = cv2.minEnclosingCircle(cnt)

          radius = int(radius)

          zeros = np.zeros((2 * radius + 10, 2 * radius + 10))
```

```
            from_x = int(max(0, x - radius))
            to_x = int(min(cols, x + radius))

            from_y = int(max(0, y - radius))
            to_y = int(min(rows, y + radius))

            h = to_x - from_x
            w = to_y - from_y

            f = int(radius - w / 2)
            t = int(radius - h / 2)

            zeros[f: f + w, t: t + h] = i[from_y:to_y, from_x:to_x]

            image = zeros
            image = cv2.resize(image, (image_size, image_size))

            image = threshold(image)

            return image

        @staticmethod
        def images(size=IMAGE_SIZE):
            """reads images from IMAGES_DIR directory and processes
them
                :return images in (id, label, image) format"""
            images = []
            i = 0
            for root, dirs, files in os.walk(IMAGES_DIR):
                for dir in dirs:
                    for root, dirs, files in os.walk(IMAGES_DIR +
dir):
                        for file in files:
                            images.append((i,                    dir,
cv2.imread(IMAGES_DIR + dir + '/' + file, cv2.IMREAD_GRAYSCALE)))
                            i += 1

            processed = []
            for id, label, image in images:
                processed.append((id,                          label,
ImageProcessor.process_image(image=image, image_size=size)))

            return processed

        @staticmethod
        def process_color_image(image):
            """Thresholds, crops and processes an image
                :arg image image in a RGB format"""
            def recreate_image(codebook, labels, w, h):
                d = codebook.shape[1]
                image = np.zeros((w, h, d))
```

```
                label_idx = 0
                for i in range(w):
                    for j in range(h):
                        image[i][j] = codebook[labels[label_idx]]
                        label_idx += 1
                return image

            def get_images(pixels, clusters_number):
                kmeans = KMeans(init='k-means++', random_state=241,
n_clusters=clusters_number).fit(pixels)
                labels = pandas.DataFrame(kmeans.predict(pixels),
columns=['C'])
                result = pandas.concat([pixels, labels], axis=1)
                mean = result.groupby(['C'])['R', 'G',
'B'].mean().as_matrix()
                mean_image = recreate_image(mean, labels.as_matrix(),
w, h)
                return mean_image

            image = img_as_float(image)
            w, h, d = image.shape
            pixels = pandas.DataFrame(np.reshape(image, (w * h, d)),
columns=['R', 'G', 'B'])

            image = get_images(pixels, 2)
            image = np.array(image, dtype=np.float32)
            image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
            image = np.array(image * 255, dtype=np.uint8)

            m = (np.min(image))
            something, image = cv2.threshold(image, m + 1, 255,
cv2.THRESH_BINARY_INV)

            # image = cv2.blur(image, (10, 10))
            kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,
(10, 10))
            image = cv2.dilate(image, kernel)
            image = cv2.erode(image, kernel)

            return ImageProcessor.process_image(image)


    class Classifier:

        spectrumData = None
        classifier = RandomForestClassifier(n_estimators=200,
n_jobs=-1)

        if USE_COLLECT_FEATURES:
            spectrumData =
pd.read_csv(SPECTRUM_FEATURES_COLLECT_FILE_PATH)
```

```
        else:
            spectrumData = pd.read_csv(SPECTRUM_FEATURES_FILE_PATH)

        shapeData = pd.read_csv(SHAPE_FEATURES_FILE_PATH)
        y = shapeData['label']
        X = spectrumData.drop(['label'], 1)
        X = collapse(X, ROWS_COLLAPSE_NUMBER)
        X = X.join(shapeData.drop(['label'], 1))
        X = X.fillna(0)

        scaler = StandardScaler()
        X = scaler.fit_transform(X)

        classifier.fit(X, y)
        classNames = list(set(y))
        classNames.sort()
        numFeatures = classifier.n_features_

        @staticmethod
        def accuracy(c=None):
            if c is None:
                c = Classifier.classifier
            generator      =      KFold(n_splits=5,      shuffle=True,
random_state=42)
            accuracy = cross_val_score(c, Classifier.X, Classifier.y,
cv=generator)

            return np.mean(accuracy)

        @staticmethod
        def classification_report(classifier):
            x_train,       x_test,       y_train,       y_test       =
train_test_split(Classifier.X,     Classifier.y,     random_state=0,
test_size=0.4)
            y_pred = classifier.fit(x_train, y_train).predict(x_test)

            return classification_report(y_test, y_pred)

        @staticmethod
        def parameters_tuning(classifier=None, grid=None):
            if grid is None:
                grid = {'n_estimators': np.arange(1, 101)}
            if classifier is None:
                classifier = Classifier.classifier
            cv = KFold(n_splits=5, shuffle=True, random_state=241)
            gs  =  GridSearchCV(classifier,  grid,  scoring='accuracy',
cv=cv, n_jobs=-1)
            gs.fit(Classifier.X, Classifier.y)
            return gs.best_estimator_, gs.cv_results_

        @staticmethod
```

```
    def show_confusion_matrix(classifier=None):
        if classifier is None:
            classifier = Classifier.classifier
        x_train,      x_test,      y_train,      y_test      =
train_test_split(Classifier.X,    Classifier.y,    random_state=0,
test_size=0.4)
        y_pred = classifier.fit(x_train, y_train).predict(x_test)

        cnf_matrix = confusion_matrix(y_test, y_pred)

        plt.figure()
        Classifier.__plot_confusion_matrix(cnf_matrix,
classes=Classifier.classNames,
                                           title='Confusion
matrix')
        plt.show()

    @staticmethod
    def predict(image):
        image = ImageProcessor.process_color_image(image)
        spectrum,            sc,            shape            =
FeatureGenerator.get_features(image)
        features_len = Classifier.classifier.n_features_
        if USE_COLLECT_FEATURES:
            spectrum = sc + ([sc[-1]] * (features_len - len(shape)
- len(spectrum)))
        else:
            spectrum  =  spectrum  +  ([0]  *  (features_len  -
len(shape) - len(spectrum)))
        spectrum = collapse(spectrum, ROWS_COLLAPSE_NUMBER)
        features = spectrum + shape
        features                                                =
Classifier.scaler.transform(np.array(features).reshape(1, -1))
        return Classifier.classifier.predict(features)

    @staticmethod
    def  __plot_confusion_matrix(cm,  classes,  title='Confusion
matrix', cmap=plt.cm.Blues):
        """
        This function prints and plots the confusion matrix.
        """
        plt.imshow(cm, interpolation='nearest', cmap=cmap)
        plt.title(title)
        plt.colorbar()
        tick_marks = np.arange(len(classes))
        plt.xticks(tick_marks, classes, rotation=270)
        plt.yticks(tick_marks, classes)
        thresh = cm.max() / 2.
        for   i,   j   in   itertools.product(range(cm.shape[0]),
range(cm.shape[1])):
            plt.text(j, i, cm[i, j],
```

```
                        horizontalalignment="center",
                        color="white" if cm[i, j] > thresh else
"black")

            plt.tight_layout()
            plt.ylabel('True label')
            plt.xlabel('Predicted label')
```